Path Planning for Vehicle Motion Control Using Numerical Optimization Methods

Ann Louise Roald

Master of Science in Cybernetics and Robotics
Submission date: June 2015
Supervisor: Morten Breivik, ITK
Co-supervisor: Anastasios Lekkas, IMT

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

The candidate will consider the problem of combining numerical optimization methods for vehicle path planning with adaptive guidance for unknown ocean current compensation. The following elements have to be considered:

1. Formulate the path planning problem for different types of vehicles using dynamic programming.
2. Implement and test numerical optimization methods to solve optimal control problems for different vehicles using numerical simulations in MATLAB.
   - The main methods to be considered are semi-Lagrangian and pseudospectral methods.
   - Considered vehicles are Dubins car, Reeds-Shepp car and simplified boat models.
3. Compare the performance of the numerical methods to get an overview of benefits and drawbacks.
4. Investigate the possibility of using these methods in combination with adaptive guidance techniques in cases where environmental disturbances are present.

Assignment given: January 12th 2015
Supervisors: Morten Breivik, ITK and Anastasios Lekkas, IMT

Preface

This thesis was carried out during the spring semester of 2015 at the Norwegian University of Science and Technology (NTNU) and concludes my Master of Science in Cybernetics and Robotics. The aim of this thesis is to develop and implement path planning systems for various underactuated vehicles using optimal control methods and numerical approximation methods.

My years at NTNU have been challenging, especially the final year where I worked independently to solve several difficult issues within advanced subjects. I have, however, learned a lot and hopefully some of my work will useful to others who choose to work with similar issues. These last five years have definitely been an unforgettable experience.

I would like to thank my supervisors at NTNU, Morten Breivik and Anastasios Lekkas, for all guidance and help during the last year. I would also like to thank my parents for moral support, encouragement and for proofreading this master's thesis.

Ann Louise Roald
Trondheim, June 7, 2015

Abstract

Path planning is an important part of many systems, especially autonomous systems. Finding the optimal paths for various vehicles, given different optimization criteria such as the shortest, fastest, straightest or energy-optimized paths, is of interest for many applications. Testing different methods gives an overview of the benefits and drawbacks so that one can choose the best solution for a specific vehicle and a specific purpose.

One aim of this project is to find the shortest paths from one pose (position and orientation) to another using optimal control theory and numerical optimization methods. In addition, guidance techniques are used to make sure that vehicles under the influence of unknown disturbances still converge towards the target. The optimal control method used is dynamic programming with the Hamilton-Jacobi-Bellman (HJB) equation. It is further investigated how to solve this using the numerical approximation methods, semi-Lagrangian approximation scheme (SL) and the pseudospectral (PS) discretization method. When implementing the Legendre pseudospectral method in this thesis, it was decided that the MATLAB application package DIDO would be used. DIDO, however, uses the Legendre pseudospectral method to discretize the optimization problem itself, not the HJB equation.

Together, these methods are used to solve the shortest path problem for three different models. The two first models utilized are both nonholonomic mobile robots, called the Dubins car (DU) and the Reeds-Shepp car (RS). The shortest path problem has previously been solved for these two models and therefore the optimal paths for these models are already known. The results from SL and PS can therefore easily be compared to the known optimal solutions. After making sure that both numerical approximation methods work for these car models, the methods and techniques are used to solve the shortest path problem for a simplified boat model, both with and without an unknown ocean current.

In the first part of this thesis, relevant theory is presented. Subsequently, the numerical approximation schemes are implemented on all models and the results are discussed and compared to give an overview of accuracy, advantages and disadvantages. Since the optimization problem is solved with the use of numerical optimization methods, the paths are optimized, not optimal. Hence, the question to be answered is how good the results are and how close they are to the optimal paths. The difficulties and advantages when using the SL scheme and PS method are investigated and discussed. The solution from SL and PS is also compared to the optimized solution given by the numerical optimization method finite difference (FD) and the optimal solution given by Pontryagin's Minimum Principle, which were both researched in a pre-project for this master's thesis. In addition, it is shown that observers and a guidance system can be used to estimate unknown disturbances, for example the ocean current, and then further be used to either minimize position error or create new optimized paths in environments with and without static obstacles.

Two main contributions pertaining to optimal control theory and numerical optimization methods are included in this thesis. The first contribution is extending the use of the semi-Lagrangian approximation scheme by using the SL scheme to identify optimized paths for boats. The second contribution is combining the numerical optimization method PS with adaptive guidance in order to conduct unknown ocean current compensation. Two different approaches were tested in order to implement this: initial path planning with continuous current estimation and repeated path planning. Both approaches were tested in environments with and without static obstacles to determine how collision avoidance is handled. Both of them enable the boat to counteract the unknown current and follow the desired optimized path given by PS. By updating the path, repeated path planning gave a shorter and more feasible path than the initial path planning approach. In this thesis, however, the implementation of repeated path planning suffers from some numerical issues that must be further investigated.

Sammendrag

Norwegian translation of the abstract

Baneplanlegging er en veldig viktig del av mange systemer, spesielt autonome systemer. Å finne en optimal bane for forskjellige fartøy, gitt optimaliseringskriterier som korteste, raskeste, retteste eller mest energisparende baner, er av stor interesse i mange applikasjoner. Ved å teste og sammenligne forskjellige metoder og få en oversikt over fordeler og ulemper, kan man enklere velge riktig metode for et spesifikt fartøy og formål.

Hovedmålet med denne oppgaven er å finne korteste bane, fra en stilling (posisjon og orientering) til en annen ved bruk av optimal reguleringsteori og numeriske optimaliseringsteknikker. Deretter vil styringsteknikker bli brukt for å sørge for at et fartøy påvirket av ukjente forstyrrelser fremdeles når sluttposisjonen. Den optimale kontrollteknikken som er brukt er dynamisk programmering ved bruk av Hamilton-Jacobi-Bellman (HJB) ligningen. Det er videre undersøkt hvordan denne kan bli løst ved hjelp av de to numeriske metodene, semi-Lagrangian (SL) og pseudospectral (PS) diskretisering. Det ble bestemt at implementasjonen av Legendre pseudospectral-metoden i denne rapporten skulle gjennomføres ved hjelp av MATLAB applikajsonspakken kalt DIDO. DIDO bruker derimot Legendre pseudospectral-metoden til å diskretisere selve optimaliseringsproblemet, ikke HJB-ligningen.

Sammen vil disse metodene bli brukt til å løse korteste vei problemet for tre forskjellige modeller. De to første modellene er begge underaktuerte mobile roboter, kalt Dubins bil (DU) og Reeds-Shepp bil (RS). Korteste vei problemet har tidligere blitt løst for begge disse modellene og deres opimale baner er derfor kjent. Resultatene fra SL og PS metodene kan dermed lett sammenlignes med den allerede kjente optimale løsningen. Etter å ha løst disse to problemene vil metodene og teknikkene bli brukt til å løse korteste vei problemet for en forenklet båtmodell, både med og uten ukjent havstrøm.

I første del av rapporten er all relevant bakgrunnsinformasjon presentert. Deretter er de numeriske metodene implementert og resultatene diskutert og sammenlignet for å gi en oversikt over de ulike løsningenes nøyaktighet, fordeler og ulemper. Siden optimaliseringsproblemet er løst ved hjelp av numeriske metoder er løsningene optimalisert, men ikke optimale. Hovedspørsmålet er da hvor bra resulatatene er og hvor nære de er de optimale løsningene. Fordeler og ulemper ved å bruke SL og PS teknikkene er også diskutert. Løsningene er i tillegg sammenlignet med løsningene gitt av metodene undersøkt i forprosjektet til denne masteroppgaven: HJB løst med den numeriske optimaliseringsmetoden finite difference (FD) og Pontryagin's Minimum Principle. Det er deretter vist at estimatorer og styringssystemer kan bli utnyttet for å estimere ukjente forstyrrelser, som for eksempel havstrøm, og at dette igjen kan brukes både for å minske posisjonsfeil og for å estimere nye optimaliserte baner, i miljøer med og uten statiske objekter.

To hovedbidrag knyttet til optimal kontrollteori og numeriske optimaliseringsteknikker er inkludert i rapporten. Den første er å utvide bruken av SL ved å identifisere optimaliserte baner for båtmodeller. Det andre bidraget er å kombinere den numeriske optimliseringsteknikken PS med adaptive styringssystemer for å kompensere for ukjent havstrøm. To ulike tilnærminger ble testet for å gjennomføre dette: initiell baneplanlegging med kontinuelig havstrømsestimering og gjentagende baneplanlegging. Begge tilnærmingene er tested i miljøer med og uten statiske objekter for å undersøke hvor bra kollisjonsunngåelse er håndtert. Begge fører til at båten klarer å motvirke den ukjente havstrømmen og dermed følge den ønskede banen kalkulert av PS. Ved å oppdatere banen underveis, ga gjentagende baneplanlegging en kortere og mer kjørbar bane enn initiell baneplanleggingstilnærmingen. I denne rapporten lider implementasjonen av gjentagende baneplanlegging av numeriske problemer som må undersøkes nærmere.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description/meaning
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
UAV	Unmanned Aerial Vehicle
AUV	Autonomous Underwater Vehicle
PMP	Pontryagin's Minimum (Maximum) Principle
HJB	Hamilton-Jacobi-Bellman
DU	Dubins car
RS	Reeds-Shepp car
GNSS	Global Navigation Satellite System
SL	Semi-Lagrangian
PS	Pseudospectral
FD	Finite Difference
FE	Forward Euler
BE	Backward Euler
H	Heun
CN	Crank-Nicolson
ENO	Essentially Nonoscillatory Interpolation
WENO	Weighted Essentially Nonoscillatory Interpolation
DDPP	Discrete Dynamic Programming Principle
VI	Value Iteration
LOS	Line-Of-Sight
DOF	Degree of Freedom

List of Symbols

Symbols	Description/meaning
$\mathbf{J}(\mathbf{x}, \mathbf{u}, t_f)$	Cost function, $\mathbf{J}[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] = \mathbf{E}(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} \mathbf{F}(\mathbf{x}(t), \mathbf{u}(t), t)\ dt$
$\mathbf{E}(\mathbf{x}(t_f), t_f)$	Terminal Cost
$\mathbf{F}(\mathbf{x}(t), \mathbf{u}(t), t)$	Running Cost
$\mathbf{e}(\mathbf{x}_f, t_f)$	Endpoint constraints
$\mathbf{H}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t)$	Hamiltonian function
m	Mass
ρ	Turning radius
ρ_{min}	Minimum turning radius
t	Time
t_f	Final time
t_0	Initial time
T	Terminal time
$\mathbf{x}^f$	Final pose
$\mathbf{x}^0$	Initial pose
$\mathbf{x}^*$	Optimal pose
K_i and c_i	Constants
$\mathbf{V}(x)$	Value function
$\Omega \subset \mathbb{R}^2$	A bounded, connected domain
Ω_{free}	Free space
Ω_{obs}	Space with obstacles
Γ	Target set, typically a final location or a final pose
$\mathscr{A}$	Admissible controls
$\mathbb{U}$	Control set
S	Straight line segment
C	Circular line segment
L	Left turn line segment
R	Right turn line segment
S^+	Straight line segment going forwards
C^+	Circular line segment going forwards
L^+	Left turn line segment going forwards
R^+	Right turn line segment going forwards
S^-	Straight line segment going backwards
C^-	Circular line segment going backwards
L^-	Left turn line segment going backwards
R^-	Right turn line segment going backwards
∇	Gradient (e.g. $\nabla_x V(\mathbf{x}, t) = \frac{\partial V}{\partial x}$)

Chapter 1

Introduction

This master's thesis concerns the development and implementation of path planning systems for motion control of underactuated vehicles by solving the shortest path problem using numerical optimization methods. The introduction presents the motivation and main contributions for this master's thesis. It also presents the structure of the thesis in order to give the reader an easy overview of the different models and techniques that are studied and implemented.

1.1 Background and Motivation

Path planning is the problem of producing one or more subpaths connecting two poses (position and orientation), and is one of the major problems to solve when it comes to autonomous dynamic systems. When finding a path between the start and finish poses it is essential to ensure that the path is executable and safe. The path being executable means that all kinematic and physical constrains must be satisfied so that it is possible for the vehicle to maintain the path. Producing safe paths includes avoiding and keeping a safe distance from obstacles [50].

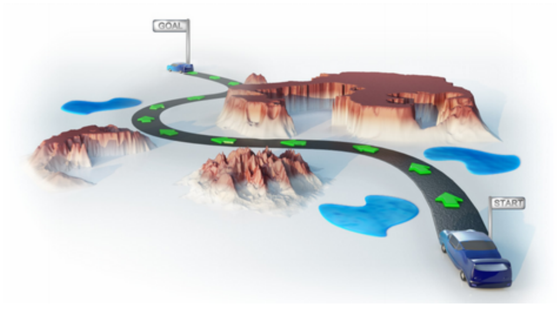


Figure 1.1: Path Planning [49]

Vehicle models are usually described by a set of ordinary differential equations (ODEs) that define the dynamics of the vehicle and the relationship between the state variables $\mathbf{x}$ and control input $\mathbf{u}$, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$. By taking into account the dynamics of the system, path constraints and bounds on state and control variables, optimal control is the problem of finding a control law such that the optimality criterion, minimizing or maximizing a cost function, is achieved [13]. Optimal control theory can be very useful in many circumstances. This theory can help provide optimal paths for various vehicles, for example, the shortest or least energy-consuming paths [50]. For more details on optimal control see [38].

The theory of the Hamilton-Jacobi-Bellman (HJB) equation is often used in optimal control problems. By considering the Dynamic Programming Principle (DPP) formulated by R. E. Bellman in the 1950s, many optimal control problems can be characterized by the HJB equation, where the associated value functions can be shown to be the unique solution of a Partial Differential Equation (PDE) [16].

Figure 1.2: Richard Ernest Bellman [51]

Optimal paths are rarely guaranteed because the HJB equation can seldom be solved analytically. When solving the problem analytically, the solution is the optimal trajectory. If this is too difficult or not possible at all, the problem can be reformulated, discretised and solved by numerical optimization. Numerical optimization methods obtain approximate optimal paths, or paths that are relatively close to the ideal optimal path [30]. Although numerical optimization methods give a solution that is not exact, the use of these methods opens up the possibility of working with more advanced and complicated models. One of the disadvantages with optimal control theory and numerical optimization techniques is that they have higher demands when it comes to the computational power required [50].

Semi-Lagrangian (SL) approximation schemes are examples of such numerical optimization methods. SL schemes provide a general method for solving dynamic programming equations [16]. One of the difficulties with this technique is that it suffers from computational complexity. This is one of the major reasons these techniques have not been used that much in the past. However, in recent years this has grown into a promising research area, both because of the improvement of computers and the development of faster and more accurate algorithms [16].

Pseudospectral (PS) methods were developed in the 1970s. At that time, they were mostly used for solving partial differential equations in fluid dynamics and meteorology [37]. First in the 1990s, PS methods were introduced for solving optimal control problems. One of the reasons for their growing popularity is that they offer computational efficiency, great accuracy, exponential convergence rate and at the same time provide Eulerian-like simplicity. The PS methods also generate a smaller scale optimization problem when compared to other methods, and therefore avoid many of the problems and difficulties experienced with the SL schemes [37]. In addition, research has uncovered that there is a close connection between the properties of the continuous time optimal control problem and the pseudospectral discrete approximation [42]. Since the PS method proved to be both very efficient and accurate, the shortest path problem using PS is further investigated in this thesis by introducing adaptive guidance techniques to estimate and remove the error caused by unknown ocean current.

1.2 Main Contributions

Two main contributions of this thesis pertaining to optimal control theory and numerical optimization methods are:

1. Extending the use of the semi-Lagrangian approximation scheme to identify optimized paths for boats
2. Combining the numerical optimization method PS with adaptive guidance in order to conduct unknown ocean current compensation, with two tested approaches:
    - initial path planning with continuous current estimation, and
    - repeated path planning

By extending the use of the semi-Lagrangian approximation scheme by using the SL scheme to identify optimized paths for boats, it is demonstrated that this scheme can be used to specify motion primitives and store optimal trajectories. Even though the SL scheme is proven to converge quite slowly, such use of the scheme is shown to be feasible.

Combining the numerical optimization method PS with adaptive guidance is demonstrated by implementing and testing two approaches in environments with and without static obstacles to determine how collision avoidance is handled. Both approaches enable the boat to counteract the unknown current and follow the desired optimized path given by PS. The first approach is to use PS to estimate one initial optimized path, assuming no ocean current. Thereafter, an adaptive guidance system and observers are used to estimate and counteract the unknown current in order to follow the desired path. The second approach is to update the optimized path at a regular time interval using PS and the current estimation given by the guidance system at that time. By updating the path, repeated path planning gives a shorter and more feasible path than the initial path planning approach. In this thesis, however, the implementation of repeated path planning suffers from some numerical issues that must be further investigated.

1.3 Overview

This master's thesis focuses on solving the shortest path planning problem by using the semi-Lagrangian approximation scheme and the pseudospectral method.

The theory explaining the basic principles of motion planning, path planning, trajectory optimization and underactuated vehicles are presented in Section 2. Section 3.1 explains the HJB equation and its properties. In Section 3.2, the basic building blocks of the different numerical schemes are described.

The numerical optimization methods researched are described in detail in sections 3.3 and 3.4, to give the reader an overview of the different elements of the schemes and their difficulties, benefits and drawbacks. In Section 4, the vehicle models used and their dynamics are presented.

After all theory is presented, the shortest path problems for the different models are solved using either the SL scheme or the PS method. The procedure and assumptions made are described and explained, and a selection of results for various initial conditions, final conditions and models are presented in sections 5 and 6. Finding the shortest path for a boat using PS is further investigated in Section 7 by introducing adaptive guidance techniques to remove the error caused by ocean current. The implementations in sections 5, 6 and 7 are all tested in environments with and without static obstacles, to determine how collision avoidance is handled.

In Section 8 the results are discussed to give an overview of how well the SL scheme and PS method worked on the different vehicle models and how close the numerical solutions were to the optimal solutions. In the conclusion, further work within this subject is also discussed.

Chapter 2

Motion Control and Vehicles

In this chapter, general theory about motion control, path planning, trajectory optimization and underactuated vehicles is presented.

2.1 Motion Control Systems

A motion control system is a system that controls the position or velocity of vehicles using actuators. An actuator is a type of motor or control surface that controls the vehicle. For a boat, examples of actuators are the propellers, tunnel and azimuth thrusters, aft rudders and stabilizing fins [17]. A motion control system is usually made up of independent blocks called guidance, navigation and control (GNC). This is illustrated in Figure 2.1. In addition, one might have a path planner. The input to the guidance system is the output of the path planner. These blocks have important and separate roles in the control system.

- **Path planner**: Uses fixed constraints based on the vehicle's characteristics, the varying constraints from the environment and the goal (start and end position or pose) to illustrate the desired results [32]. The output of the path planner can be waypoints, desired trajectory, etc.

- **Guidance**: Takes in the output of the path planner and data provided by the navigation system. The output from the path planner is used as "instructions" for where to go. Then the guidance system continuously computes the reference position, velocity and acceleration the vehicle must have to be able to fulfill the instructions from the path planner [32]. The trajectory from the path planner is not the same as the reference trajectories from the guidance system. The trajectory from the guidance system takes into account both the trajectory or waypoints from the path planner and the vehicle's current position, before calculating a reference trajectory the control system should follow. This is seen in Figure 2.1.

- **Navigation**: Typically uses GNSS (global navigation satellite system) and motion sensors (accelerometer, gyroscopes) to determine the vehicle's position, velocity, acceleration, attitude and distance [32].

- **Control**: Uses information from the navigation system and reference trajectory from the guidance system to calculate the necessary control forces to follow the reference trajectory as closely as possible. These control forces are the output of the control system and the input to the actuators [17].

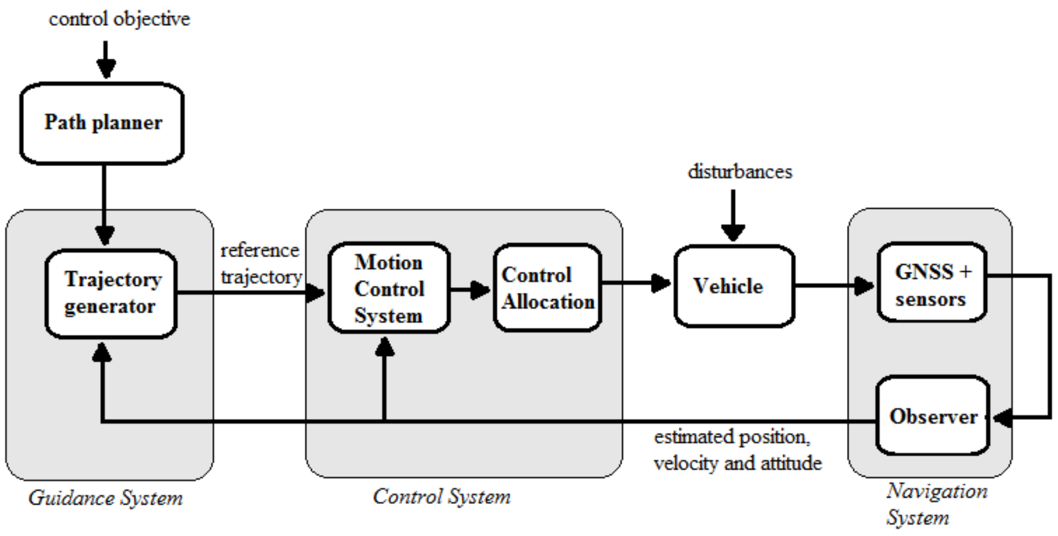


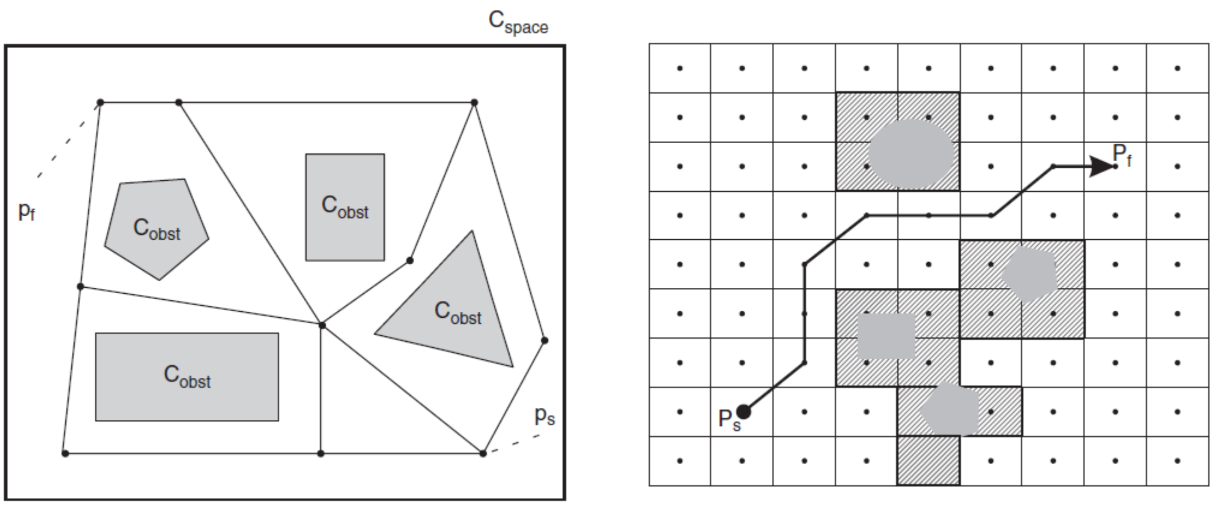
Figure 2.1: Motion Control System. Consists of a path planner, guidance system, control system and navigation system. These systems combined provide the nessecary control forces needed to acheive the control objective (modified from [17]).

2.2 Path Planning

Path planning is the problem of producing one or more paths connecting initial and final position or pose, and is one of the major problems to solve when it comes to autonomous mobile systems. Often the entire path from initial to final pose consists of many consecutive paths that connect different points of interest [50]. These points may be places the vehicle must go through to fulfill its purpose, for example, taking depth measurements or photographs at specific locations. In other cases the points may be constructed as a safety measure in order to avoid obstacles in the nearby environment.

When finding a path between the start and finish poses, or between two subsequent points of interest, it is very important to consider the constraints of the system. Two of the most important constraints to take into account are the constraints that make sure the path is feasible and the constraints that make sure the path is safe [50]. A path is considered feasible when all kinematic and physical constrains are satisfied, therefore allowing the vehicle to maintain the path. This includes, for example, minimum turning radius. Producing safe paths includes avoiding obstacles and keeping a safe distance to them. With no further constraints on a path, the path planner will in most cases produce several different path possibilities.

Various techniques can be used to produce routes between two points. When referring to a route planner, one usually refers to a system that produces a series of nodes that are connected by straight lines and starts and stops at the start and finish points. This is also called a global path planner, and produces routes for the given map of pre-known locations. The locations can be places to go to or places to avoid, like obstacles, no-fly areas and threats to avoid. Since the route planner produce routes consisting of straight-line segments, the routes cannot be followed by the vehicle directly [50]. However, these techniques are very useful since they produce these paths faster than the time required to produce a drivable path. It is therefore fairly common for a path planner to first produce a route and then later to refine the route such that it becomes a feasible path.



(a) Road Map Method (b) Cell Decomposition Method

Figure 2.2: Route Planning Techniques. Producing nodes connected by straight lines [50]

Many of the techniques used to produce routes come from the field of robotics (see [9]) and some of the most well-known are the road map method, the cell decomposition method and the potential field method. The road map and cell decomposition methods (illustrated in Figure 2.2) transform the environment by discretizing the map into small areas or cells, while the potential field method transforms the environment into a continuous function. Thereafter, a search algorithm is used to find a path connecting the start and target points [50]. Another family of path planning techniques is optimal control or trajectory optimization, which are very useful techniques used when solving problems with additional objective functions, constraints and boundary conditions. Some trajectory optimization techniques can give feasible paths directly, but the dimension and complexity of solving these optimal control problems will in general lead to a much higher computational overhead. It can also be difficult to solve path planning problems by using optimal control techniques in an environment with many or large obstacles [50].

2.3 Trajectory Optimization

Trajectory optimization pertains to the methods for designing a trajectory or path that, in addition to being safe and feasible, minimize or maximize some measure of performance. Some examples are minimizing path length, time, energy and curvature [13].

One of the most common ways to illustrate an optimal control problem is:

Minimize $\mathbf{J}[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] = \mathbf{E}(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} \mathbf{F}(\mathbf{x}(t), \mathbf{u}(t), t)\ dt$

Subject to

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$$

$$\mathbf{g}_i(\mathbf{x}, \mathbf{u}, t) \leq 0, \qquad i = 1, ..., m \tag{2.1}$$

$$\mathbf{h}_i(\mathbf{x}, \mathbf{u}, t) = 0, \qquad i = 1, ..., p \tag{2.2}$$

where $\mathbf{J}[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f]$ is the objective to be minimized, $\dot{\mathbf{x}}$ is the dynamics of the system, $\mathbf{g}_i(\mathbf{x}, \mathbf{u}, t)$ are the inequality constraints and $\mathbf{h}_i(\mathbf{x}, \mathbf{u}, t)$ are the equality constraints. The constraints of the problem can be anything from accepted control input values to a predefined endpoint position.

There are many different techniques available to solve optimization problems, where the solutions are found by either analytic or numerical procedures. Whether the problem can be solved analytically or numerically, closely relates to whether the solution is optimal or just optimized. Optimal paths are rarely guaranteed because optimal path problems can rarely be solved analytically. When solving the problem analytically the solution is the optimal trajectory. If this for some reason is too difficult or not possible at all, the problem can be reformulated, discretised and solved by numerical optimization. Numerical optimization methods obtain approximate optimal paths, or paths that are relatively close in the spatial domain to the ideal optimal path. In this case the solution is not optimal, only optimized [30]. How close the optimized solution is to the optimal solution depends on the transformations and approximations used. In conclusion, methods and techniques based on optimal control theory, where the aim is to find the optimal path, will often result in optimized paths because they are solved using numerical approximations.

2.4 Underactuated Vehicles

Underactuation is a technical term often used in robotics and control theory and it is very important to distinguish between an underactuated vehicle and a fully actuated vehicle. An underactuated vehicle is a vehicle that cannot follow an arbitrary trajectory in the configuration space.

Definition 1. *(Configuration Space) Configuration space is the space of possible positions and orientations that a vehicle may obtain [17].*

Underactuation can occur for different reasons, but the most common reason in control theory is that the vehicle cannot produce control forces and moments in all degrees of freedom (DOF). DOF is the set of displacements and rotations that together describes the change of a vehicle's position and orientation [17]. For a vehicle in 3D space the maximum amount of DOFs is 6, surge, sway and heave (u, v, w) to describe position changes and roll, pitch and yaw, (p, q, r), to describe orientation changes. This can be seen in Figure 4.4. A normal car, for example, is underactuated because of the nonholonomic constraints given by the wheels, which cause the car to be unable to accelerate in a direction perpendicular to the direction the wheels are facing. While controlling a fully actuated vehicle is trivial, controlling and producing feasible paths for an underactuated vehicle is more complex. An underactuated vehicle often has limitations on what control objectives they are able to fulfill [17].

In this thesis, underactuated and fully actuated vehicles are therefore defined as [17]:

Definition 2. *(Underactuated Vehicle) A vehicle is underactuated if it has fewer control inputs than generalized coordinates.*

Definition 3. *(Fully Actuated Vehicle) A vehicle is fully actuated if it has equal or more control inputs than generalized coordinates.*

Chapter 3

Hamilton-Jacobi-Bellman Equation and Numerical Optimization

In this chapter, the general idea behind the Hamilton-Jacobi-Bellman equation and how this equation can be used to solve the optimal control problem is presented. The more specific HJB equations for the minimum time and shortest path problems are also presented. Two numerical schemes that can be used for solving an optimal control problem numerically are also described in detail.

3.1 Hamilton-Jacobi-Bellman Equation

The Hamilton-Jacobi-Bellman (HJB) equation is a partial differential equation (PDE) and is based on theory of dynamic programming. The solution of the HJB equation is called the value function $V(\mathbf{x}, t)$. The value function, defined as $V(x) = \inf\limits_{u \in A} J(\mathbf{x}, \mathbf{u}, t)$, is also called the cost-to-go function and gives the minimum cost for a problem that consists of the ODEs for the model and the cost function to minimize $J(\mathbf{x}, \mathbf{u}, t)$. In this thesis the value function gives the minimum costs of the paths between two poses, where the lowest cost represents the shortest path to take. When solved over the entire state space, the HJB equation gives both necessary and sufficient conditions for the optimal path [31]. In other words, one can produce the HJB equation from the optimization problem. The problem of solving the entire system of equations is then reduced to solving one single partial differential equation [23].

The dynamics of a system is often represented as:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad && 0 < t < T \\ \mathbf{x}(t_0) &= \mathbf{x}_0 && \end{aligned}$$

where $\mathbf{x}$ is the state of the system, $\mathbf{u} : (t_0, T) \longrightarrow A \subseteq \mathbb{R}^m$ is the control function, and $\mathbf{f} : \mathbf{R} \times (t_0, T) \times A \longrightarrow \mathbf{R}^d$ is the controlled drift or dynamics [16]. When the admissible controls are defined as $\mathscr{A} := \{\mathbf{u} : (t_0, T) \to A, \text{measurable}\}$, then for any control $\mathbf{u} \in \mathscr{A}$ the solution is well defined (in a weak sense). The cost function, defined as $\mathbf{J}[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] = \mathbf{E}(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} \mathbf{F}(\mathbf{x}(t), \mathbf{u}(t), t)\ dt$, will then be used to find the most optimal of these solutions. $\mathbf{E}(\mathbf{x}(t_f), t_f)$ represents the terminal cost and $\mathbf{F}(\mathbf{x}(t), \mathbf{u}(t), t)$ the running cost. The terminal cost depends only on the terminal time and pose, while the running cost varies with state, control action and time [16].

By reformulating this problem into many subsequent similar problems, but with different start times and poses, the dynamics of the system is represented as [15]:

$$\begin{aligned} &\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) && t_0 < t < T \\ &\mathbf{x}(t_0) = \mathbf{x}^{t_0} \end{aligned} \tag{3.1}$$

By evaluating (3.1) for all starting times between 0 and T and for every starting pose $\mathbf{x}^{t_0} \in \mathbb{R}^n$, the value function $V(\mathbf{x}, t)$ is found.

By using the dynamic programming principle, the Hamilton-Jacobi-Bellman equation [15] is defined as:

$$\begin{cases} V_t(\mathbf{x}, t) + \max\limits_{u \in A}\{\mathbf{f}(\mathbf{x}, \mathbf{u})\nabla_x V(\mathbf{x}, t) + \mathbf{F}(\mathbf{x}, \mathbf{u}, t)\} = 0, & (\mathbf{x}, t) \in \mathbf{R}^d \times (0, T) \\ V(\mathbf{x}, T) = \mathbf{E}(\mathbf{x}(t_f), t_f), & \mathbf{x} \in \mathbf{R}^d \end{cases} \tag{3.2}$$

If the value function's first derivatives are continuous then $V(\mathbf{x}, t)$ solves the HJB equation. This is proven in [15]. The value function is a representation of the best value obtained from the cost function and further assists in constructing the optimal solution [16].

In [15] C. E. Lawrence describes a list of steps which should be followed when using dynamic programming to find the optimal control. These are:

1. Solve the HJB equation and use this to calculate the value function, $V(\mathbf{x}, t)$.
2. For each point, $\mathbf{x} \in \mathbb{R}^n$ and each time $0 < t < T$ define $\mathbf{u}(\mathbf{x}, t) = \mathbf{c}$ to be the parameter value that maximizes the HJB equation.
3. Solve the ODE in (3.1) with $\mathbf{u}(\mathbf{x}, t)$ (find $\mathbf{x}^*(t)$).
4. Define the feedback controller $\mathbf{u}^*(t) := \mathbf{u}(\mathbf{x}^*(t), t)$.

The optimal controls found when using dynamic programming and the HJB equation are often referred to as optimal feedback controls. These are expressed as functions of the state of the system. As a result, these controls are able to account for errors in the real state of the system caused by, for example, model errors or external perturbations. The counterpart to feedback controls is open-loop controls, which are functions of time only [16]. One typical method that calculates open-loop controls is Pontrygain's Minimum Principle (PMP).

A comparison between the HJB method and PMP was studied in the pre-project for this master's thesis. The main conclusion from the pre-project was that Pontryagin's Minimum Principle gives necessary conditions, while the Hamilton-Jacobi-Bellman equation gives necessary and sufficient conditions that a path must fulfill to be optimal. Therefore, HJB gives conditions which, if fulfilled, will give one unique global solution for every situation. HJB is also a more flexible method and can be expanded to work for more advanced and complicated models. The HJB equation can, however, rarely be solved analytically, thus the problem must often be solved using numerical optimization techniques.

3.1.1 The Minimum Time and Shortest Path Problems

The general setup for an optimal control problem, with the only constraints being a start and a finish pose, is often represented as:

Minimize/Maximize $\mathbf{J}[\mathbf{x}(\cdot),\mathbf{u}(\cdot),t_f] = \mathbf{E}(\mathbf{x}(t_f),t_f) + \int_{t_0}^{t_f} \mathbf{F}(\mathbf{x}(t),\mathbf{u}(t),t)\ dt$
Subject to

$$\begin{aligned}&\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})\\&\mathbf{x}(0) = \mathbf{x}^0\\&\mathbf{x}(t_f) = \mathbf{x}^f\end{aligned}$$

where $\dot{\mathbf{x}}$ represents the dynamics of the system, $\mathbf{x}(0)$ and $\mathbf{x}(t_f)$ the initial and final conditions and $\mathbf{J}[\mathbf{x}(\cdot),\mathbf{u}(\cdot),t_f]$ the cost function to be minimized or maximized.

For the shortest path problem, $\mathbf{J} = \int_{t_0}^{t_f} \sqrt{\dot{x}^2+\dot{y}^2}$ for a 2D problem and $\mathbf{J} = \int_{t_0}^{t_f} \sqrt{\dot{x}^2+\dot{y}^2+\dot{z}^2}$ for a 3D problem. For a minimum time optimization problem, J becomes $\mathbf{J} = t_f$. When setting the speed of, for example, the Dubins car to be constant and equal to 1, the cost function for a shortest path problem will equal the minimum time problem, $\mathbf{J} = \int_{t_0}^{t_f} \sqrt{\dot{x}^2+\dot{y}^2} = \int_{t_0}^{t_f} \sqrt{1} = \int_{t_0}^{t_f} 1 = t_f$.

The HJB equations for these optimal control problems become (3.3) for the shortest path problem and (3.4) for the minimum time problem.

$$\begin{cases} V_t(\mathbf{x},t) + \max\limits_{u\in A}\{\mathbf{f}(\mathbf{x},\mathbf{u})\nabla_x V(\mathbf{x},t) + \sqrt{\dot{x}^2+\dot{y}^2} = 0, \quad (\mathbf{x},t)\in\mathbf{R}^d\times(0,T)\\ V(\mathbf{x},T) = 0\end{cases} \tag{3.3}$$

$$\begin{cases} V_t(\mathbf{x},t) + \max\limits_{u\in A}\{\mathbf{f}(\mathbf{x},\mathbf{u})\nabla_x V(\mathbf{x},t) + 1 = 0, \quad (\mathbf{x},t)\in\mathbf{R}^d\times(0,T)\\ V(\mathbf{x},T) = 0\end{cases} \tag{3.4}$$

3.2 Numerical Optimization Methods

When it is difficult or impossible to find the exact solution, numerical optimization methods are used to find numerical approximations to the solutions [25]. There are relatively few differential equations that can be solved analytically. For some practical purposes like control theory, the numerical approximations to the solution are still sufficiently accurate.

One important element of numerical methods is the discretization of the equations. After the discretization other techniques are used to calculate the solution from the descretized equations. Discretization is the process of dividing a continuous function into a finite number of discrete elements. A structured grid is often used to store the local discrete values. When discretizing an equation it is very important to find accurate replacements that maintain the global information and structure of the original problem [24]. Two examples of numerical optimization methods currently used are the finite difference (FD) method and the semi-Lagrangian approximation scheme, the latter of which is one of the methods researched in this thesis.

The FD method uses one of the simplest forms of discretization. The main principle is that the derivatives of the PDE are, at each grid point, replaced by an incremental ratio [24]. The value V_j^n is the approximated value of $V(x_j, t_n)$ at node (x_j, t_n). For the time derivative a forward increment ratio is used

$$V_t(x,t) \approx \frac{V(x, t+\Delta t) - V(x,t)}{\Delta t} \tag{3.5}$$

$$V_t(x_j, t_n) \approx \frac{V_j^{n+1} - V_j^n}{\Delta t} \tag{3.6}$$

There are several choices for the space derivative when using finite difference approximation. The choice affects the convergence of the numerical solution. An upwind discretization uses the time derivative shown above in combination with the left or right increment ratio. A central discretization uses the time derivative and the centered finite difference space derivative [24].

$$V_x(\mathbf{x}_j, t_n) \approx \frac{V_{j+1}^n - V_j^n}{\Delta x} \qquad \text{(right finite difference)} \tag{3.7}$$
$$V_x(\mathbf{x}_j, t_n) \approx \frac{V_j^n - V_{j-1}^n}{\Delta x} \qquad \text{(left finite difference)} \tag{3.8}$$
$$V_x(\mathbf{x}_j, t_n) \approx \frac{V_{j+1}^n - V_{j-1}^n}{2\Delta x} \qquad \text{(centered finite difference)} \tag{3.9}$$

Another way to approximate the space derivative is to use a directional derivative as seen in (3.10). By combining this with the time derivative of the finite difference approximation and defining δ as Δt, the semi-Lagrangian (SL) scheme shown in (3.11) is obtained.

$$cV_x(\mathbf{x}_j, t_n) \approx \frac{V^n(\mathbf{x}_j - c\delta) - V_j^n}{\delta} \tag{3.10}$$

$$V_j^{n+1} = V^n(\mathbf{x}_j - c\Delta t) \tag{3.11}$$

The SL scheme was first proposed by Courant, Isaacson and Rees in 1952 in [10]. One of the main differences between the FD method and the SL scheme is that the FD method only iterates along the grid nodes, while the SL scheme uses directional points, $(x_j - c\delta)$, which coordinates are most often between the grid points.

The pseudospectral method uses another approach to discretize the equations. The PS method used in this thesis is also a grid-based method. The grid points are, however, chosen at certain locations called Chebyshev nodes, which are distributed over the interval [-1, 1]. The transformation between the physical domain $\tau \in [\tau_0, \tau_f]$ and the computational domain $t \in [-1, 1]$ is:

$$\tau(t) = \frac{(\tau_f - \tau_0)t + (\tau_f - \tau_0)}{2} \tag{3.12}$$

Thereafter, the value function can be approximated by [46]

$$\boldsymbol{V}_N(t) = \sum_{l=0}^{n} \boldsymbol{v}_l \phi_l(t) \tag{3.13}$$

where $\phi_l(t)$ are orthogonal functions, for example, Fourier integrals, Legendre polynomials or Chebyshev polynomials [27].

The partial derivatives can be approximated by [46]

$$\frac{\partial \boldsymbol{V}_N(t)}{\partial j} = \sum_{l=0}^{n} \boldsymbol{v}_l \frac{\partial \phi_l(t)}{\partial j} \tag{3.14}$$

Both the SL scheme and the PS method are described in more detail in sections 3.3 and 3.4. In general, when using numerical optimization methods, errors of approximation or rounding occur. It is important to have some idea of their nature and magnitude. It is important not to introduce large errors that ruin the output of the numerical approximation. On the other hand, having high accuracy might be computationally expensive. It is therefore important to know the tolerances in each system so that a reasonable trade-off between required computation resources, solution accuracy and timeliness of the answer can be found.

3.3 Semi-Lagrangian Scheme

The SL scheme was first proposed as a first-order scheme by Courant, Isaacson and Rees in 1952 in [10].

The Lagrangian and Eulerian approaches are two different ways of looking at motion. When looking at a fluid parcel in motion, the Lagrangian specification of the flow field is where each fluid parcel is followed as it moves, thereby giving the pathline by plotting the position of this specific parcel through time. The Eulerian approach, on the other hand, looks at the fluid motion but focuses on specific locations where the fluid passes instead of at specific parcels [4]. These specifications are sometimes denoted as the Lagrangian and Eulerian frame of reference. Because of this difference in the reference frame, a fully Lagrangian scheme uses a grid that moves with the flow, while an Eulerian scheme uses a fixed grid.

The semi-Lagrangian scheme uses a fixed grid, but the advective derivatives are calculated in a Lagrangian frame. The other spatial derivatives are calculated in an Eulerian frame. As a result, the advective terms can be discretized in a Lagrangian perspective and then the Lagrangian solution can be transformed back onto an Eulerian grid using interpolation to estimate the value at specific grid points. This technique avoids both the instability of Eulerian schemes and the complications and computer problems of fully Lagrangian schemes [4].

In addition to the SL descretization mentioned in Section 3.2, there are several other elements that need to be constructed to be able to solve an HJB equation by using an SL scheme [16]. One important element is a strategy for moving along the characteristics, also called approximation schemes. As mentioned in previous sections, the SL scheme uses directional derivatives. Therefore, another important element is a technique to reconstruct numerical solutions that are not at any specific grid node. This is called interpolation. Both of these elements are explained in more detail in Section 3.3.1 and Section 3.3.2.

3.3.1 Reconstruction Techniques

The role of the reconstruction element in SL schemes is to calculate the value of the numerical solution at the feet of the characteristics, which for an SL scheme usually is not at a specific grid point. SL schemes, therefore, do not use cell averages, but instead reconstruction based on the interpolation of pointwise values of the solution [16]. For explanations below, it is assumed that there is a grid with N nodes for each state in $\mathbf{x}$, and that the value function $V(\mathbf{x})$ is represented at each node. To reconstruct $V(\mathbf{x})$ a strategy is used. Some reconstruction strategies that can be used are Essentially Nonoscillatory interpolation (ENO), Weighted Essentially Nonoscillatory interpolation (WENO), finite element interpolation and symmetric Lagrangian interpolation [16]. In this thesis, it was decided that symmetric Lagrangian interpolation would be used.

Symmetric Lagrangian Interpolation

When using symmetric Lagrangian interpolation, the samples of V used for reconstruction are taken from the surrounding nodes. Lagrangian interpolation is often implemented with evenly spaced nodes and an odd degree, r, of the interpolation. When implementing with an odd degree the same number of nodes is used on both sides of the point, thus resulting in a symmetric behaviour [16].

There are several ways of implementing a Lagrangian interpolation. As mentioned, there can be different degrees on the interpolation. The most common are linear, cubic and quintic interpolation, which are equivalent to degree 1, 3 and 5, respectively. The number of nodes on each side used for these interpolation degrees is $\frac{r+1}{2}$. This is seen in Figure 3.1 for linear, cubic and quintic interpolation.

The general form for the Lagrangian interpolation is [16]:

$$I_r[V](x) = \sum_{x_k \in \mathscr{I}} V_k \psi_k(x) = \sum_{x_k \in \mathscr{I}} V_k \prod_{x_i \in \mathscr{I} \setminus \{x_k\}} \frac{x - x_i}{x_k - x_i} \tag{3.15}$$

where the stencil of nodes is defined as: $\mathscr{I} = \left\{x_k : l - \frac{r-1}{2} \leq k \leq l + \frac{r+1}{2}\right\}$

Another extension to consider is how to deal with a system that has multiple dimensions. One way to solve this issue is by splitting the operation into separate interpolations in each dimension. For two dimensions, x_1 and x_2, this equals [16]:

$$I_r[V](\mathbf{x}) = \sum_{j_1} \sum_{j_2} V(x_{j_1,j_2}) \psi_{j_1}(x_1) \psi_{j_1}(x_2) = \sum_{j_2} \Big[\sum_{j_1} V(x_{j_1,j_2}) \psi_{j_1}(x_1) \Big] \psi_{j_1}(x_2) \tag{3.16}$$

where $\psi_{j_i} = \prod_{x_i \in \mathscr{I} \setminus \{x_k\}} \frac{x - x_i}{x_k - x_i}$.

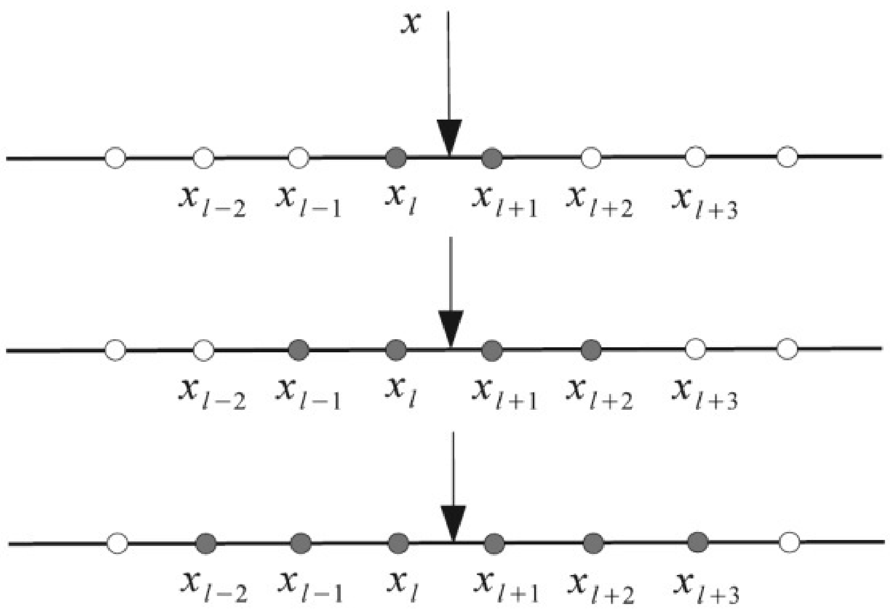


Figure 3.1: Interpolation Stencil for Linear, Cubic and Quintic Interpolation [16]

3.3.2 Approximation Schemes

Both when calculating the value function and when calculating the optimal trajectory, the value to be propagated is found by moving along the characteristics. This element is therefore to approximate a system of ODE's over a single timestep, Δt [16]. Both one-step and multi-step schemes can be used. The main difference between these two types of schemes is that one-step schemes only use one previous value of the numerical solution to approximate the subsequent value, while multi-step methods use more than one previous value. Multi-step schemes are therefore more complex, but they may often achieve greater accuracy as they use more of the pre-known information about the solution than one-step methods do [12]. In this thesis, one-step schemes are used.

One-step Schemes

In a one-step scheme, the discretized system is on the form:

$$\begin{cases} y_{k+1} = y_k + \Delta t \Phi(\Delta t; y_k, t_k, y_{k+1}), \\ y_0(x_0, t_0) = x_0, \end{cases} \tag{3.17}$$

The four main one-step schemes are Forward Euler (FE), Backward Euler (BE), Heun (H) and Crank-Nicolson (CN) [16]. The differences between these schemes is the choice of $\Phi(\Delta t; y_k, t_k, y_{k+1})$. This can be seen in Table 3.1. The FE and BE schemes are the simplest to understand and implement, but in some cases they can be inaccurate. The H and CN schemes are more complex, but since they use information from both the previous and next step to estimate the new value, they can produce more accurate calculations.

Scheme	$\Phi(\Delta t; y_k, t_k, y_{k+1})$	Order
FE	$f(y_k, t_k)$	p = 1
BE	$f(y_{k+1}, t_k + \Delta t)$	p = 1
H	$\frac{1}{2}\Big[f(y_k, t_k) + f(y_k + \Delta t f(y_k, t_k), t_k + \Delta t)\Big]$	p = 2
CN	$\frac{1}{2}\Big[f(y_k, t_k) + f(y_{k+1}, t_k + \Delta t)\Big]$	p = 2

Table 3.1: One-step Schemes, from [16]

3.3.3 Solving Optimal Control Problems

In order to solve the HJB equation numerically, the discrete version must first be found by introducing a time step, Δt. The general finite horizon discrete HJB equation then becomes [11]

$$V^{k+1}(\mathbf{x}) = \min_{u \in A} \Big\{ e^{-\lambda \Delta t} V^k(\mathbf{x} + \Delta t f(\mathbf{x}, u)) + \Delta t \mathbf{F}(\mathbf{x}(t), \mathbf{u}(t), t) \Big\}$$

To solve the numerical equation, a structured grid of our subdomain Q containing the target Γ must be built. The grid consists of $\mathbf{x}_i$ nodes, where $i \in I := \{1, ..., N\}$, with a distance $\Delta \mathbf{x}$ between each node.

When working with control problems, it is useful to classify each node as one of the subsets presented below [11].

$$\begin{aligned} I_\Gamma &= \{ i \in I : x_i \in \Gamma \} \\ I_{in} &= \{ i \in I \setminus I_\Gamma : \text{ there exists } u \in A \text{ such that } x_i + \Delta t f(x_i, u) \in Q \} \\ I_{out} &= \{ i \in I \setminus I_\Gamma : x_i + \Delta t f(x_i, u) \notin Q \} \end{aligned}$$

With these subsets, it is easier to build a fully discrete scheme specifically made for optimal control problem. The value function for the finite horizon optimal control problem, is then updated by the following rules.

$$V_i^{k+1} := \begin{cases} \min_{u \in A} \Big\{ e^{-\lambda \Delta t} V^k(\mathbf{x} + \Delta t f(\mathbf{x}, u)) + \Delta t \mathbf{F}(\mathbf{x}(t), \mathbf{u}(t), t) \Big\}, & i \in I_{in} \\ 0, & i \in I_\Gamma \\ 5000, & i \in I_{out} \end{cases}$$

In this thesis, V^0 is initialized as:

$$V_i^0 = \begin{cases} 0, & i \in I_\Gamma \\ 5000, & \text{elsewhere} \end{cases}$$

With V_i^{k+1} and V^0 described above, V is guaranteed a monotone decreasing convergence to the fixed point V^* [11]. By using this algorithm the information flows from the target and to the other nodes in the grid. In the first iteration the values in the neighbourhood of the target Γ will decrease. Then at the next iteration the values in the neighbourhood of the target's neighbourhood will decrease, and so on.

Minimum Time and Shortest Path Problems

Two examples of optimal control problems are the minimum time problem and the shortest path problem. They are very similar except for the cost function for the shortest path problem being

$$\mathbf{J}[\mathbf{x}(\cdot),\mathbf{u}(\cdot),t_f] = \int_0^{t_f} \sqrt{\dot{x}^2+\dot{y}^2}dx \tag{3.18}$$

and for the minimum time problem

$$\mathbf{J}[\mathbf{x}(\cdot),\mathbf{u}(\cdot),t_f] = t_f \tag{3.19}$$

It can be seen that with constant speed equal to 1, the cost function for the shortest path is reduced to the minimum time cost function.

In Section 3.1.1, the dynamics and HJB equations for the minimum time problem and shortest path problem were defined. In this section, the fully discrete scheme consisting of both the discrete dynamic programming principle (DDPP) and the space discretization for both problems are shown.

When using Kruzov change of variable, $V(\mathbf{x}) = 1 - e^{\lambda \Delta t}$, on the minimum time problem the DDPP is [16]:

$$\begin{cases} V^{\Delta t}(\mathbf{x}) = \min\limits_{u\in A}\Big\{e^{-\Delta t}V^{\Delta t}(\mathbf{x}+\Delta t f(\mathbf{x},u))\Big\} + 1 - e^{-\Delta t}, & \mathbf{x}\in \mathbf{R}^d \setminus \Gamma \\ V^{\Delta t}(\mathbf{x}) = 0, & \mathbf{x}\in\Gamma \end{cases} \tag{3.20}$$

By including the space discretization described in Section 3.2, the fully discrete scheme for the minimum time problem is [16]:

$$\begin{cases} V_j^{k+1} = e^{-\Delta t}\min\limits_{u\in A}\Big\{I[V^k](\mathbf{x}+\Delta t f(\mathbf{x},u))\Big\} + 1 - e^{-\Delta t}, & \mathbf{x}\in \mathbf{R}^d \setminus \Gamma \\ V_j^{k+1} = 0, & \mathbf{x}\in\Gamma \end{cases} \tag{3.21}$$

For the shortest path problem, the DDPP and space discretization give a fully discrete scheme represented by:

$$\begin{cases} V_j^{k+1} = \min\limits_{u \in A} \Big\{ e^{-\Delta t} I[V^k](\mathbf{x} + \Delta t f(\mathbf{x}, u)) + \Delta t \sqrt{\dot{x}^2 + \dot{y}^2} \Big\}, & \mathbf{x} \in \mathbf{R}^d \setminus \Gamma \\ V_j^{k+1} = 0, & \mathbf{x} \in \Gamma \end{cases} \tag{3.22}$$

The step-by-step algorithm for these two specific problems is shown in Algorithm 1 and Algorithm 2.

Algorithm 1: Value Iteration for Minimum Time Optimal Control Problem, from [1]

```
Data: Mesh G, $\Delta t$, initial guess $V^0$, tolerance $\epsilon$
while $\|V^{k+1} - V^k\| \geq \epsilon$ do
    forall the $x_i \in I_{in}$ do
        $V_i^{k+1} = \min_{a \in A} \Big\{ e^{-\Delta t} I[V^k](x_i + \Delta t f(x_i, a)) + 1 - e^{-\Delta t} \Big\}$;
    forall the $x_i \in I_\Gamma$ do
        $V_i^{k+1} = 0$;
    forall the $x_i \in I_{out}$ do
        $V_i^{k+1} = 5000$ ;
    $k = k + 1$;
```

Algorithm 2: Value iteration for Shortest Path Otimal Control Problem

```
Data: Mesh G, $\Delta t$, initial guess $V^0$, tolerance $\epsilon$
while $\|V^{k+1} - V^k\| \geq \epsilon$ do
    forall the $x_i \in I_{in}$ do
        $V_i^{k+1} = \min_{u \in A} \Big\{ e^{-\Delta t} I[V^k](\mathbf{x} + \Delta t f(\mathbf{x}, u)) + \Delta t \sqrt{\dot{x}^2 + \dot{y}^2} \Big\}$;
    forall the $x_i \in I_\Gamma$ do
        $V_i^{k+1} = 0$;
    forall the $x_i \in I_{out}$ do
        $V_i^{k+1} = 5000$ ;
    $k = k + 1$;
```

3.4 Pseudospectral Method

The Pseudospectral method is an emerging numerical technique that has been successfully used to solve partial differential equations [46], like the HJB equation. PS methods were first developed in the 1970s. At that time they were mostly used for solving partial differential equations in fluid dynamics and meteorology. First in the 1990s, PS methods were introduced to solve optimal control problems. One of the reasons for their growing popularity is that they offer computational efficiency, great accuracy and exponential convergence rate, while also providing Eulerian-like simplicity. The PS methods also generate a much smaller-scale optimization problem when compared to other methods, thereby avoiding many of the problems and difficulties of the SL scheme [37]. In addition, research has uncovered that there is a close connection between the properties of the continuous time optimal control problem and the pseudospectral discrete approximation [42].

Spectral methods, in general, are a class of techniques used to numerically solve differential equations. These methods differ from many of the traditional discretization methods because they focus on the approximation of the tangent bundle instead of the differential equation. They also achieve what is called spectral accuracy, which is arrived at when using a global approach. As a result, all the available information of a space-dependent field is used to estimate the derivative at a point [45].

PS methods are closely related to spectral methods, but they include a pseudospectral basis as well in order to represent functions on a quadrature grid. The pseudospectral method is therefore solved pointwise in physical space like the finite difference and the SL scheme. The state derivatives, however, are calculated like spectral methods, using orthogonal functions, for example, Fourier integrals, Legendre polynomials or Chebyshev polynomials. They are subsequently evaluated using either matrix-matrix multiplications, Fourier transform (FFT) or convolutions [27].

3.4.1 DIDO: A MATLAB Application Package

DIDO is a MATLAB application package used to solve optimal control problems based on pseudospectral optimal control theory [26]. Although the pseudospectral method can be used to discretize the HJB equation, DIDO solves the optimal control problem by using the Legendre pseudospectral discretization method described in Section 3.4.2 to transform the problem into an equivalent discretisized optimization problem, and then solves this using a well-developed Nonlinear Program (NLP) solver [26]. The NLP solver used is based on Sequential Quadratic Programming (SQP) and called SNOPT (described in detail in [21]).

3.4.2 Legendre Pseudospectral Approximations

The Legendre pseudospectral method, which is one version of a pseudospectral method for solving optimal control problems, uses the basic discretization principles of pseudospectral methods. In simple terms, the Legendre pseudospectral method is just another way of discretizing the problem to cast a smooth nonlinear optimization problem. It does this using Legendre polynomials and Gauss Lobatto quadratures [26]. The same discretization can be used to discretize, for example, the HJB equation.

The goal of this PS method is to solve the original optimization problem B by discretizing it to problem B^N in a way that permits the discretized problem to transform with dualization. This transformation requirement is also called the covector mapping principle. A covector mapping principle, shown in Figure 3.2, shows the connection between dualization and discretization. This means that any optimal solution of the discretized problem B^N must also satisfy the necessary conditions of the discretized dualized problem $B^{\lambda N}$. This way of solving optimal control problems is far simpler than developing and solving for the necessary conditions [22].

The initial problem B is represented by:

Minimize $\mathbf{J}[\mathbf{x}(\cdot), \mathbf{u}(\cdot), \tau_f] = \mathbf{E}(\mathbf{x}(\tau_f), \tau_f) + \int_{t_0}^{t_f} \mathbf{F}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau)\ d\tau$

Subject to

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, \tau)\\ \mathbf{g}(\mathbf{x}, \mathbf{u}, \tau) &\leq 0\\ \mathbf{h}(\mathbf{x}, \mathbf{u}, \tau) &= 0\end{aligned}$$

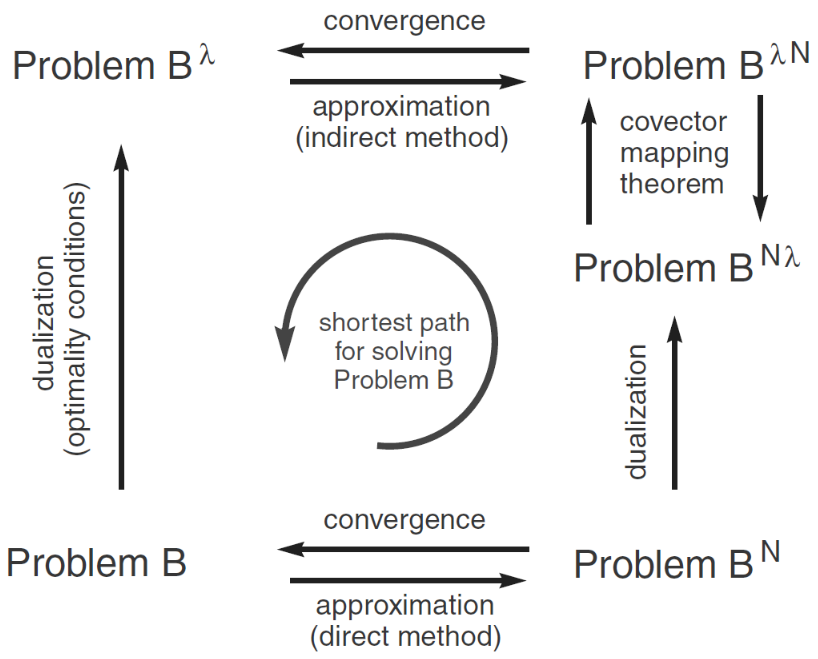


Figure 3.2: Covector Mapping Principle, from [22]

By using the approximation and techniques from [45] described below, the initial problem B is discretized into problem B^N.

In the Legendre PS method, the grid points are the shifted Legendre–Gauss–Lobatto (LGL) points at which the shift is achieved by mapping the physical domain $\tau \in [\tau_0, \tau_f]$ to a computational domain $t \in [-1, 1]$ using the affine transformation:

$$\tau(t) = \frac{(\tau_f - \tau_0)t + (\tau_f - \tau_0)}{2} \tag{3.23}$$

The state and control functions are approximated by Nth degree polynomials [45]:

$$\boldsymbol{x}(\tau(t)) \approx \boldsymbol{x}^N(\tau(t)) = \sum_{l=0}^{n} \boldsymbol{x}_l \phi_l(t) \tag{3.24}$$

$$\boldsymbol{u}(\tau(t)) \approx \boldsymbol{u}^N(\tau(t)) = \sum_{l=0}^{n} \boldsymbol{u}_l \phi_l(t) \tag{3.25}$$

where for l = 0, 1,..., N

$$\phi_l(t) = \frac{1}{N(N+1)L_n(t_l)} \frac{(t^2-1)\dot{L_n}(t)}{t-t_l} \tag{3.26}$$

are the Lagrange interpolating polynomials of order N and L_N is the Legendre polynomial of degree N. It can be proven that

$$\phi_l(t) = \delta_{lk} = \begin{cases} 1 & \text{if l = k} \\ 0 & \text{if } l \neq k \end{cases} \tag{3.27}$$

Therefore $\boldsymbol{x}_l = \boldsymbol{x}^N(\tau_l)$, $\boldsymbol{u}_l = \boldsymbol{u}^N(\tau_l)$ where $\tau_l = \tau(t_l)$ so that $\tau^N \equiv \tau_f$. By differentiating (3.24) and evaluating it at the node points, t_k, the result is:

$$\dot{\boldsymbol{x}}^N(\tau_k) = \frac{d\boldsymbol{x}^N}{d\tau}\bigg|_{\tau=\tau_k} = \frac{d\boldsymbol{x}^N}{dt}\frac{dt}{d\tau}\bigg|_{t_k} = \frac{2}{\tau_f-\tau_0}\sum_{l=0}^{N} D_{kl}\boldsymbol{x}_{kl} \equiv \frac{2}{\tau_f-\tau_0}\boldsymbol{d}_k \tag{3.28}$$

where $D_{kl} = \dot{\phi}_l(t_k)$ are entries of the (N +1)x(N +1) differentiation matrix, called D. This differentiation matrix is represented as:

$$\boldsymbol{D} := [D_{kl}] = \begin{cases} \frac{L_n(t_k)}{L_N(t_l)}\frac{1}{t_k-t_l} & l \neq k \\ -\frac{N(N+1)}{4} & \text{l = k = 0} \\ \frac{N(N+1)}{4} & \text{l = k = N} \\ 0 & \text{otherwise} \end{cases} \tag{3.29}$$

This causes the approximation of the state dynamics to become:

$$\frac{\tau_f-\tau_0}{2}\boldsymbol{f}(\boldsymbol{x}_k,\boldsymbol{u}_k) - \sum_{l=0}^{N} D_{kl}\boldsymbol{x}_l = 0 \qquad k = 0,...,N \tag{3.30}$$

Then the cost function, $\boldsymbol{J}$, is approximated using the Gauss-Lobatto integration rule.

$$\boldsymbol{J}[\mathbf{X}^N,\mathbf{U}^N,\tau_0,\tau_f] = E(\boldsymbol{x}_o,\boldsymbol{x}_N,\tau_0,\tau_f) + \frac{\tau_f-\tau_0}{2}\sum_{l=0}^{N} F(\boldsymbol{x}_k,\boldsymbol{u}_k)w_k \tag{3.31}$$

where

$$\mathbf{X}^N = [\boldsymbol{x}_0;\boldsymbol{x}_1;...;\boldsymbol{x}_N], \qquad \mathbf{U}^N = [\boldsymbol{u}_0;\boldsymbol{u}_1;...;\boldsymbol{u}_N] \tag{3.32}$$

and w_k are the LGL weights given by

$$w_k := \frac{2}{N(N+1)}\frac{1}{[L_N(t_k)]^2}, \qquad k = 0,1,...,N \tag{3.33}$$

By putting all this together, problem B is discretized into the nonlinear programming (NLP) problem, named B^N:

Find the $(N+1)(N_x+N_u)+2$ vector $\mathbf{X}_{NP} = (\mathbf{X}^N; \mathbf{U}^N; \tau_0; \tau_f)$ that:

Minimize $\mathbf{J}(\mathbf{X}_{NP}) \equiv \mathbf{J} = \mathbf{E}(\boldsymbol{x}_0, \boldsymbol{x}_N, \tau_0, \tau_f) + \frac{\tau_f - \tau_0}{2} \sum_{l=0}^{N} F(\boldsymbol{x}_k, \boldsymbol{u}_k) w_k$

Subject to

$$\begin{aligned}
&\frac{\tau_f - \tau_0}{2} \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) - \sum_{l=0}^{N} D_{kl} \boldsymbol{x}_l = 0 \\
&\mathbf{g}(\mathbf{x}, \mathbf{u}, \tau) \leq 0 \\
&\mathbf{h}(\mathbf{x}, \mathbf{u}, \tau) = 0
\end{aligned}$$

Similarly, as the state and control functions are approximated, the HJB equation could be discretized by [46]

$$\boldsymbol{V}_N(t) = \sum_{l=0}^{n} \boldsymbol{v}_l \phi_l(t) \tag{3.34}$$

Although the PS methods have many advantages like high accuracy and computational savings, they also have some drawbacks. One of the drawbacks is that constraints are only enforced at the LGL node points and at the specific event times. As a result, the constraints are not always satisfied between these node points [26]. This may cause problems if there are too few nodes. One example of this is that the calculated path may cut the corner of an obstacle because the nodes on either side of it are in the clear space, thus resulting in infeasible paths.

Chapter 4

Vehicle Models

This chapter presents the various vehicle models studied in this thesis by looking into the system dynamics, problem formulation and difficulties with solving specific optimization problems.

4.1 Car Models

A nonholonomic mobile robot is a car model that includes the two principle kinematic constraints of a normal car, which are rolling without slipping and rolling wheels with directional bound. The latter constraint makes sure that the vehicle only moves along the steering direction, tangential to the main axis [2]. As a result, the vehicle can not move for example sideways, like a boat or airplane can. Although this simplifies the differential equations of the model, it also makes the path planning problem more difficult. With these wheeled robots the aim in this thesis is to optimize the path length and therefore find the shortest path each vehicle uses to get from one pose to another. For both cases the cost function is

$$\mathbf{J}[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] = \int_0^{t_f} \sqrt{\dot{x}^2 + \dot{y}^2} dx \tag{4.1}$$

where t_f is the final time. In the following section, the system dynamics of two types of nonholonomic mobile robots are described: The Dubins car and the Reeds-Shepp car. In both cases, the dynamic variables of the system are $\mathbf{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ and the system dynamics are $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. The optimization problem is stated as:

Minimize $\mathbf{J}[\mathbf{x}(\cdot), u(\cdot), t_f] = \int_0^{t_f} \sqrt{\dot{x}^2 + \dot{y}^2} dx$
Subject to

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{x}(0) &= \mathbf{x}^0 \\ \mathbf{x}(t_f) &= \mathbf{x}^f \end{aligned}$$

4.1.1 Dubins Car

The Dubins car is a car-like model that is assumed to have constant speed, v, and constant minimum turning radius. Minimum turning radius is defined as $\rho_{min} = \frac{L}{\tan(\phi)}$, where L is the length between the front and rear axles and ϕ is the steering angle [31].

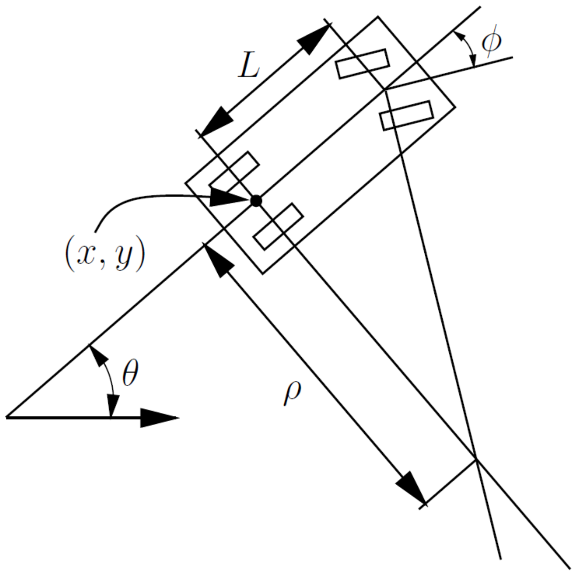


Figure 4.1: Dubins Car, from [31]

The motion of the car [31] can therefore be represented by (4.2).

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0$$
$$\Downarrow$$
$$\begin{aligned} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \frac{v}{L}\tan(\phi) \end{aligned} \tag{4.2}$$

By setting the constant speed and minimum turning radius equal to 1, (4.2) can be simplified to

$$\dot{x} = \cos(\theta) \tag{4.3}$$
$$\dot{y} = \sin(\theta) \tag{4.4}$$
$$\dot{\theta} = \frac{u}{\rho_{min}} \tag{4.5}$$

where u $\in [-1, 1]$. This is the input to the steering wheel that determines whether the car turns right, $-1 < u < 0$, left, $0 < u < 1$, or goes straight ahead, $u = 0$. Dubins car can be further extended to include altitude. The model is then called Dubins airplane [8].

Dubins Optimal Paths

Dubins path refers to the shortest curve that connects two poses (position and orientation) in the two-dimensional Euclidean plane, the x-y plane. In 1957, Lester Eli Dubins proved by using geometry [14] that the shortest path always consists of maximum curvature turns and/or straight line segments [50]. For the Dubins car the optimal paths were proven to be one of the following 6 types [31]: $L_\alpha R_\beta L_\gamma, R_\alpha L_\beta R_\gamma, L_\alpha S_d L_\gamma, L_\alpha S_d R_\gamma, R_\alpha S_d L_\gamma, R_\alpha S_d R_\gamma$, where $\alpha, \gamma \in [0, 2\pi)$, $\beta \in (\pi, 2\pi)$, and $d \geq 0$. Some of these path types are illustrated in Figure 4.2. It is, however, in some applications not always possible to find a Dubins path. This was shown in [48].

This concatenation will, however, in reality lead to difficult manoeuvres between the straight and circular segments. The transition causes a jump in the curvature because straight segments have a curvature of $\kappa = 0$ and circle segments have a curvature of $\kappa = \frac{1}{\text{radius}}$. As a result, a sudden change in lateral acceleration occurs, which can lead to deviations from the Dubins path. This problem was addressed in [3], where continuous curvature paths with a upperbounded derivative were designed [32].

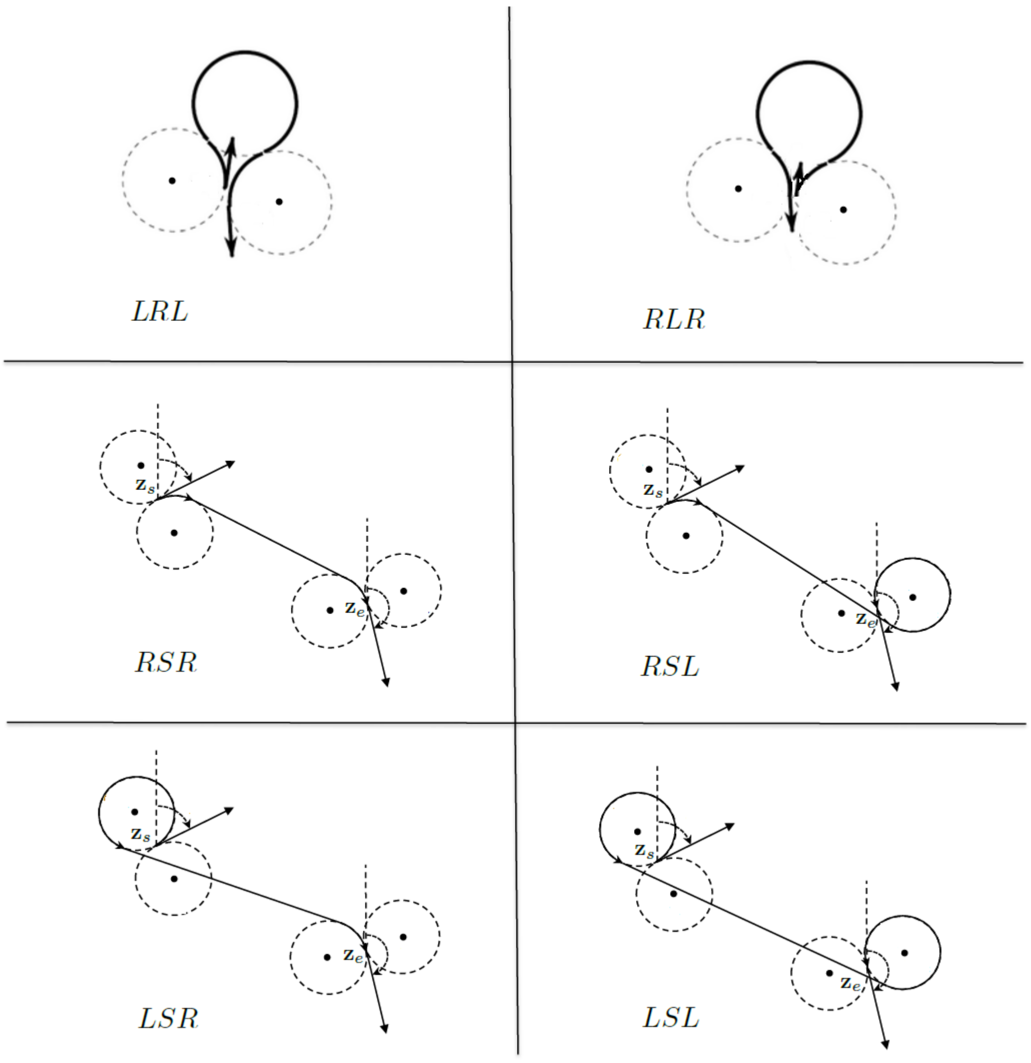


Figure 4.2: Optimal Paths for Dubins Car: CSC and CCC. Edited from [28] and [40]

4.1.2 Reeds-Shepp Car

The Reeds-Shepp car has dynamics similar to those of the Dubins car. The only difference is that the Reeds-Shepp car does not have constant velocity, v. Instead, $|v| = k$ and the car can therefore alternate between driving forwards, v = k, and backwards, v = -k. This opens up a lot of new feasible paths. By choosing k = 1 the dynamics for this vehicle becomes [31]

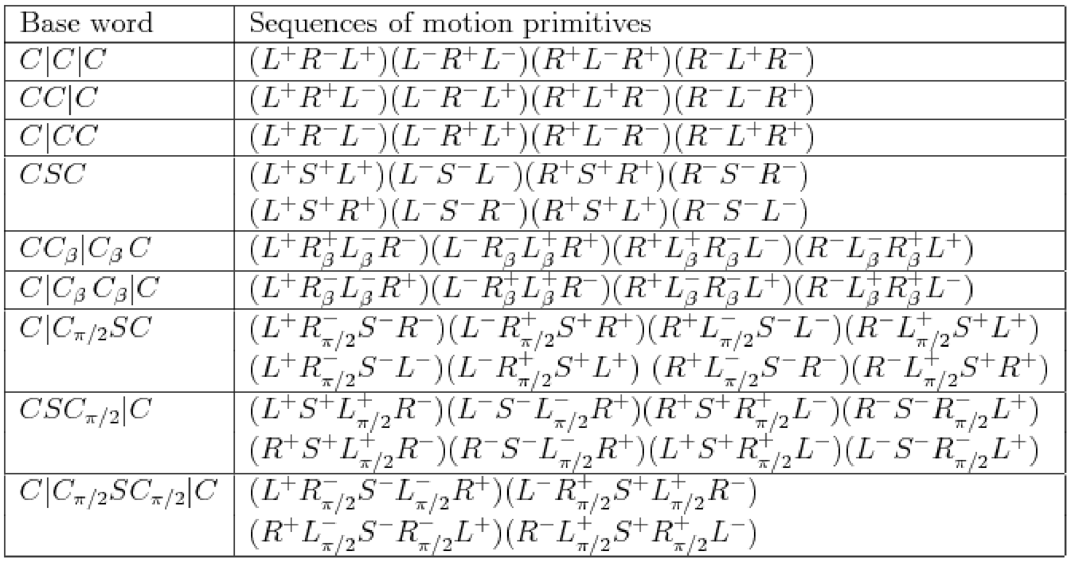
$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0$$
$$\Downarrow$$

$$\dot{x} = u_1 \cos(\theta) \tag{4.6}$$
$$\dot{y} = u_1 \sin(\theta) \tag{4.7}$$
$$\dot{\theta} = \frac{u_2}{\rho_{min}} \tag{4.8}$$

where $u_1 \in \{-1, 1\}$ is the control input that determines whether the car goes forwards or backwards, and $u_2 \in [-1, 1]$ is the control input that determines whether the car goes straight or makes a turn to the left or right.

Reeds-Shepp Optimal Paths

J. A. Reeds and L. A. Shepp showed in [43], using advanced calculus, that the shortest path for a Reeds-Shepp car was one of the 48 types shown in Figure 4.3. All of the paths consist of different path segments: Driving straight, turning left or turning right with minimum turning radius. S denotes straight, R right, L left and C circular turn. C then includes both L and R. In addition, + denotes driving forwards and - denotes driving backwards. The subscriptions on some of the path segments stand for the duration of these segments.



Base word	Sequences of motion primitives
$C\|C\|C$	$(L^+R^-L^+)(L^-R^+L^-)(R^+L^-R^+)(R^-L^+R^-)$
$CC\|C$	$(L^+R^+L^-)(L^-R^-L^+)(R^+L^+R^-)(R^-L^-R^+)$
$C\|CC$	$(L^+R^-L^-)(L^-R^+L^+)(R^+L^-R^-)(R^-L^+R^+)$
CSC	$(L^+S^+L^+)(L^-S^-L^-)(R^+S^+R^+)(R^-S^-R^-)$ $(L^+S^+R^+)(L^-S^-R^-)(R^+S^+L^+)(R^-S^-L^-)$
$CC_\beta\|C_\beta C$	$(L^+R^+_\beta L^-_\beta R^-)(L^-R^-_\beta L^+_\beta R^+)(R^+L^+_\beta R^-_\beta L^-)(R^-L^-_\beta R^+_\beta L^+)$
$C\|C_\beta C_\beta\|C$	$(L^+R^-_\beta L^-_\beta R^+)(L^-R^+_\beta L^+_\beta R^-)(R^+L^-_\beta R^-_\beta L^+)(R^-L^+_\beta R^+_\beta L^-)$
$C\|C_{\pi/2}SC$	$(L^+R^-_{\pi/2}S^-R^-)(L^-R^+_{\pi/2}S^+R^+)(R^+L^-_{\pi/2}S^-L^-)(R^-L^+_{\pi/2}S^+L^+)$ $(L^+R^-_{\pi/2}S^-L^-)(L^-R^+_{\pi/2}S^+L^+)\ (R^+L^-_{\pi/2}S^-R^-)(R^-L^+_{\pi/2}S^+R^+)$
$CSC_{\pi/2}\|C$	$(L^+S^+L^+_{\pi/2}R^-)(L^-S^-L^-_{\pi/2}R^+)(R^+S^+R^+_{\pi/2}L^-)(R^-S^-R^-_{\pi/2}L^+)$ $(R^+S^+L^+_{\pi/2}R^-)(R^-S^-L^-_{\pi/2}R^+)(L^+S^+R^+_{\pi/2}L^-)(L^-S^-R^-_{\pi/2}L^+)$
$C\|C_{\pi/2}SC_{\pi/2}\|C$	$(L^+R^-_{\pi/2}S^-L^-_{\pi/2}R^+)(L^-R^+_{\pi/2}S^+L^+_{\pi/2}R^-)$ $(R^+L^-_{\pi/2}S^-R^-_{\pi/2}L^+)(R^-L^+_{\pi/2}S^+R^+_{\pi/2}L^-)$

Figure 4.3: Reeds-Shepp Paths, from [31]

4.2 Boat Models

The kinematics of a boat is often represented using six independent coordinates, u (surge), v (sway), w (heave), p (roll), q (pitch) and r (yaw), which describe change in position and orientation. The first three are used to describe position and translation, while the last three are used to describe orientation and rotational motion [17]. This is seen in Figure 4.4.

There are many different ways to model a boat. A boat can, for example, be considered to have one actuator (rudder) if constant speed is assumed, or two actuators (rudder and thrust) if there is varying speed. In addition, boat models are often modelled in 2D assuming zero pitch and roll and no change in heave position. One example of the kinematics of a boat model with one actuator in 2D is [41]:

$$\dot{x} = \cos(\psi)u - \sin(\psi)v \tag{4.9}$$

$$\dot{y} = \sin(\psi)u + \cos(\psi)v \tag{4.10}$$

$$\dot{\psi} = r \tag{4.11}$$

where x, y and ψ give the position and orientation of the ship in the earth-fixed frame. The control input for this model is yaw, the angular velocity around the z-axis. In (4.9), u is the velocity in the surge direction and v is the velocity in the sway direction. Surge and sway velocity are modelled as:

$$\dot{u} = v\frac{r}{\rho} + \frac{1}{m}(u_d - u) \tag{4.12}$$

$$\dot{v} = -u\frac{r}{\rho} - vd \tag{4.13}$$

where d is the damping constant and u_d the desired surge speed.

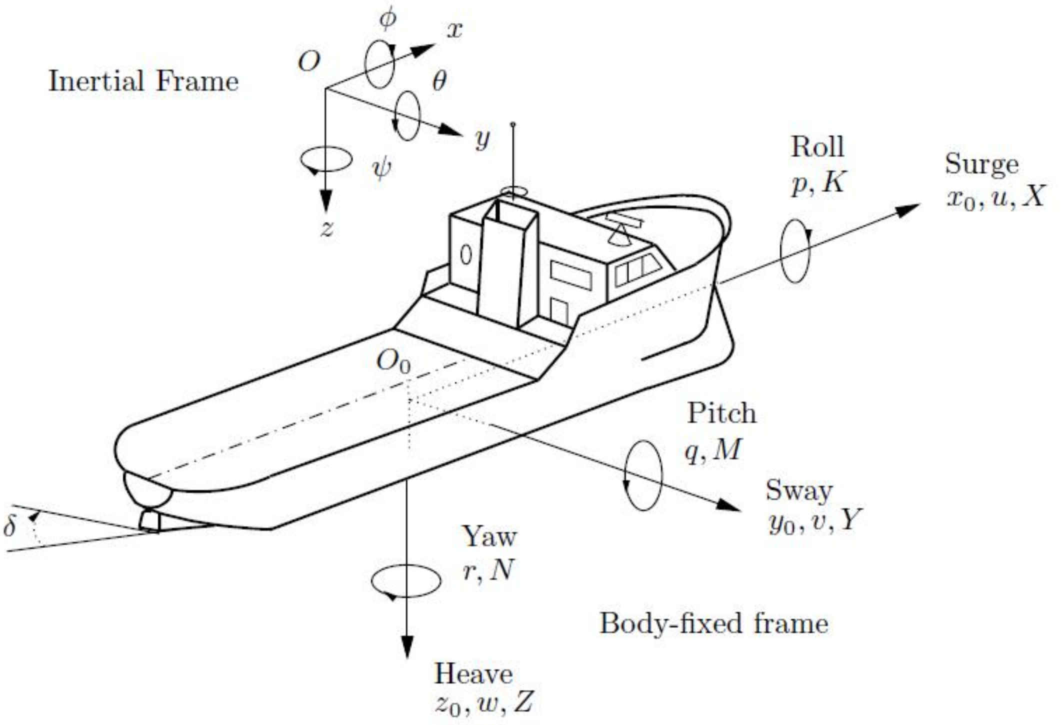


Figure 4.4: Ship - Degrees of Freedom [39]

By assuming that the sway velocity is very small, it can be neglected. This results in a very simplified boat model with the following boat kinematics:

$$\dot{x} = \cos(\psi)u \tag{4.14}$$

$$\dot{y} = \sin(\psi)u \tag{4.15}$$

$$\dot{\psi} = r \tag{4.16}$$

In addition, boats are always affected by the ocean that surrounds them. The path of a boat will therefore always be effected by the ocean current. These currents are often unknown. By adding this to the boat model above, the model becomes:

$$\dot{x} = \cos(\psi)u + V_x \tag{4.17}$$
$$\dot{y} = \sin(\psi)u + V_y \tag{4.18}$$
$$\dot{\psi} = r \tag{4.19}$$

where V_x and V_y are the unknown currents that affect the position of the boat, in x and y directions. V_x and V_y will both be constant in a NED frame.

The same can be done for the model including sway velocity. Then the full model becomes:

$$\dot{x} = \cos(\psi)u - \sin(\psi)v + V_x \tag{4.20}$$
$$\dot{y} = \sin(\psi)u + \cos(\psi)v + V_y \tag{4.21}$$
$$\dot{\psi} = r \tag{4.22}$$
$$\dot{u} = v\frac{r}{\rho} + \frac{1}{m}(u_d - u) \tag{4.23}$$
$$\dot{v} = -u\frac{r}{\rho} - vd \tag{4.24}$$

When introducing the terms relative surge velocity, $u_r = u - u_c$, and relative sway velocity, $v_r = v - v_c$, the models become:

$$\dot{x} = \cos(\psi)u_r - \sin(\psi)v_r + V_x \tag{4.25}$$
$$\dot{y} = \sin(\psi)u_r + \cos(\psi)v_r + V_y \tag{4.26}$$
$$\dot{\psi} = r \tag{4.27}$$
$$\dot{u_r} = v_r\frac{r}{\rho} + \frac{1}{m}(u_{rd} - u_r) \tag{4.28}$$
$$\dot{v_r} = -u_r\frac{r}{\rho} - v_r d \tag{4.29}$$

where u_c and v_c are the body-fixed ocean current velocities. The body-fixed ocean current velocities are related to the North-East (NED) ocean current velocities by $[u_c, v_c]^T = \boldsymbol{R}^T(\psi)[V_x, V_y]^T$. Finally, u_{rd} is the relative desired surge speed.

Chapter 5

Semi-Lagrangian Implementation and Results

In this chapter the design, implementation and results using the SL scheme for the different vehicle models described in Chapter 4 are presented. In this thesis, SL is implemented with a forward Euler approximation scheme and linear interpolation.

5.1 Dubins Car Model

This section describes the calculations and results when solving the Hamilton-Jacobi-Bellman equation by using the SL scheme to produce the minimum time path for a Dubins car. The theory used to conduct the implementation is found in sections 3.1, 3.3 and 4.1.

By using the model represented by (4.3) - (4.5) and the general HJB equation for a minimum time problem shown in (3.4), the following HJB equation needs to be solved:

$$\begin{cases} V_t(\mathbf{x},t) + \max\limits_{u\in[-1,1]}\{V_x(\mathbf{x},t)cos(\theta) + V_y(\mathbf{x},t)sin(\theta) + V_\theta(\mathbf{x},t)\dfrac{u}{\rho_{min}} + 1 = 0 \\ V(\mathbf{x},T) = 0 \end{cases}$$

This is solved using the steps described in Section 3.3.3. A detailed description of the steps in the implementation is also shown in Algorithm 1.

A selection of the minimum time paths given by this implementation is shown in figures 5.1 to 5.4. The corresponding test values and run conditions are shown in Table 5.1. In these figures the green dot represents the starting point while the red dot represents the target point. In Figure 5.5 the value functions are illustrated together with the optimized paths. For some paths it may look like the car misses its target by some small distance. The reason for this is a combination of the discretization used and the use of an expanded target. Using an expanded target will in some cases make the algorithm converge faster. In the value function plots, the expanded target is represented by a white square around the target.

Initial pose	Final pose	N	Domain Ω	$h_x/h_y, h_\theta$	ρ_{min}	Figure
$(1, 1, \frac{3\pi}{2})$	$(-1, 1, \frac{\pi}{2})$	40	$[-2, 2] \times [0, 2\pi)$	0.1, 0.1571	1	5.1
$(1, 1, \frac{3\pi}{2})$	$(-1, -1, \frac{3\pi}{2})$	40	$[-2, 2] \times [0, 2\pi)$	0.1, 0.1571	1	5.2
$(1, 1, \pi)$	$(-1, -1, \frac{5\pi}{4})$	40	$[-2, 2] \times [0, 2\pi)$	0.1, 0.1571	1	5.3
$(1, -1, \frac{5\pi}{4})$	$(-1, 1, \frac{\pi}{4})$	40	$[-2, 2] \times [0, 2\pi)$	0.1, 0.1571	0.5	5.4

Table 5.1: Test Values for Optimized Paths for Dubins Using SL Scheme

After implementing the minimum time paths for the Dubins car in an obstacle-free environment, the implementation was expanded to include obstacles the car had to avoid. To achieve this, obstacles had to be created and defined as a part of the subset $I_{out} = \{i \in I \setminus I_\Gamma : x_i + \Delta t f(x_i, u) \notin Q\}$. This is described in Section 3.3.3. Extra code was also added to make sure the value function $\mathbf{V}$ was not updated at the nodes within the subset I_{out}, hence keeping their initialized values.

In figures 5.6 and 5.7, optimized paths for the Dubins car in an environment including obstacles are shown. The corresponding test values and run conditions are shown in Table 5.2. Two different comparisons are made. In the first comparison, the optimized paths are calculated using a different number of nodes to illustrate the effect the number of nodes have on the resulting path. The second comparison is for two different obstacle shapes. The shapes are very similar but one has a rounded corner while the other does not. It is shown in these plots that the path is affected by even a slight change in the shape of the obstacles. Figure 5.8 shows an example of a path in an environment with two obstacles. All of the value functions for these tests are illustrated together with the optimized paths in Figure 5.9. Another issue appearing both with and without obstacles is the oscillation in the control input. When examining the control inputs in figures 5.1 to 5.4, 5.6 and 5.7, it can be seen that the car switches between steering directions more than it should when compared to the optimal Dubins paths, as described in 4.1.1. The small steering mistakes can be explained by the discretization. In addition, the optimized control input sometimes switches back and forwards between left (u_1 = 1) and right turn (u_1 = -1), instead of going straight forward (u_1 = 0). This seems to be a bigger issue in an environment with obstacles.

Initial pose	Final pose	N	Domain Ω	ρ_{min}	$h_x/h_y, h_\theta$	Figure
$(1.5, -1.25, \pi)$	$(0, 1.5, \frac{\pi}{2})$	30	$[-2, 2] \times [0, 2\pi)$	0.5	0.1333, 0.2094	5.6a
$(1.5, -1.25, \pi)$	$(0, 1.5, \frac{\pi}{2})$	60	$[-2, 2] \times [0, 2\pi)$	0.5	0.0667, 0.1047	5.6b, 5.7a and 5.7b
$(4.5, -2.5, \frac{\pi}{2})$	$(-0.5, -1.75, \frac{\pi}{2})$	60	$[-5, 5] \times [0, 2\pi)$	1	0.1667, 0.1047	5.8

Table 5.2: Test Values for Optimized Paths for Dubins with Obstacles Using SL Scheme

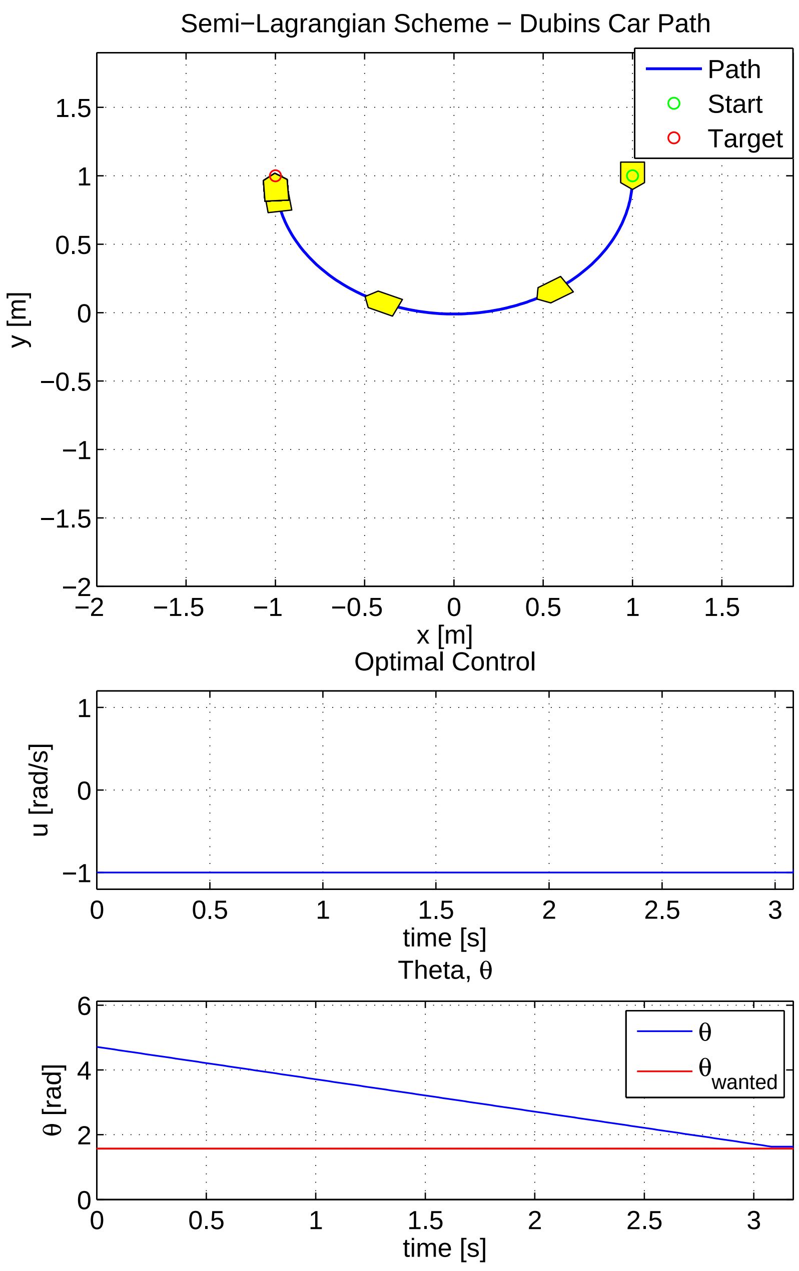


Figure 5.1: Dubins Car Using SL - Optimized Path from $(1, 1, \frac{3\pi}{2})$ to $(-1, 1, \frac{\pi}{2})$

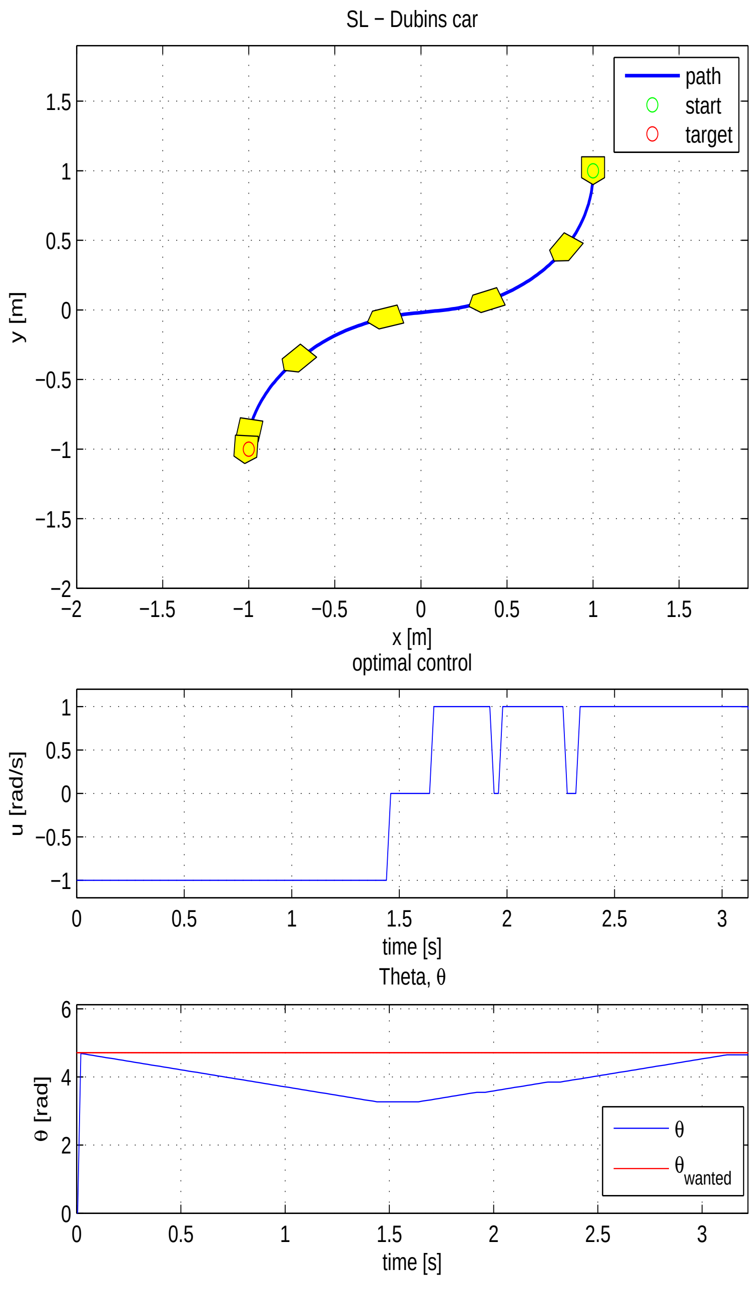


Figure 5.2: Dubins Car Using SL - Optimized Path from $(1, 1, \frac{3\pi}{2})$ to $(-1, -1, \frac{3\pi}{2})$

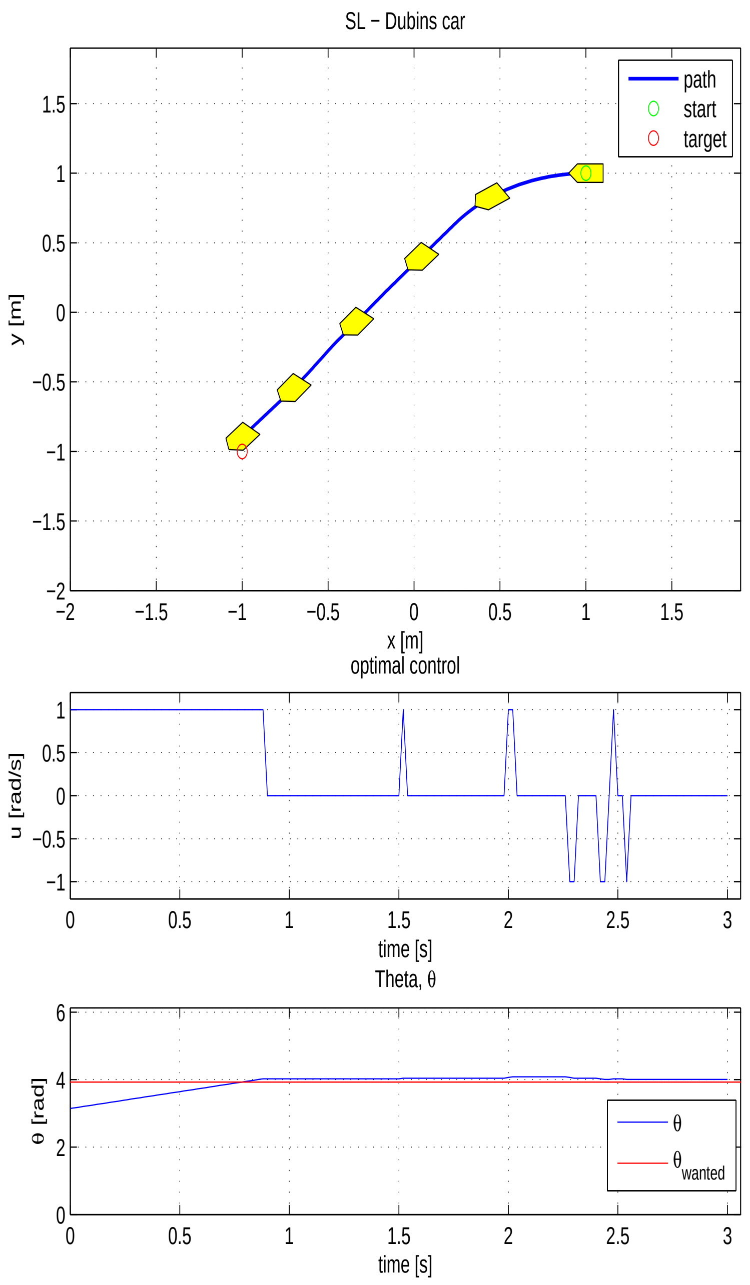


Figure 5.3: Dubins Car Using SL - Optimized Path from $(1, 1, \pi)$ to $(-1, -1, \frac{5\pi}{4})$

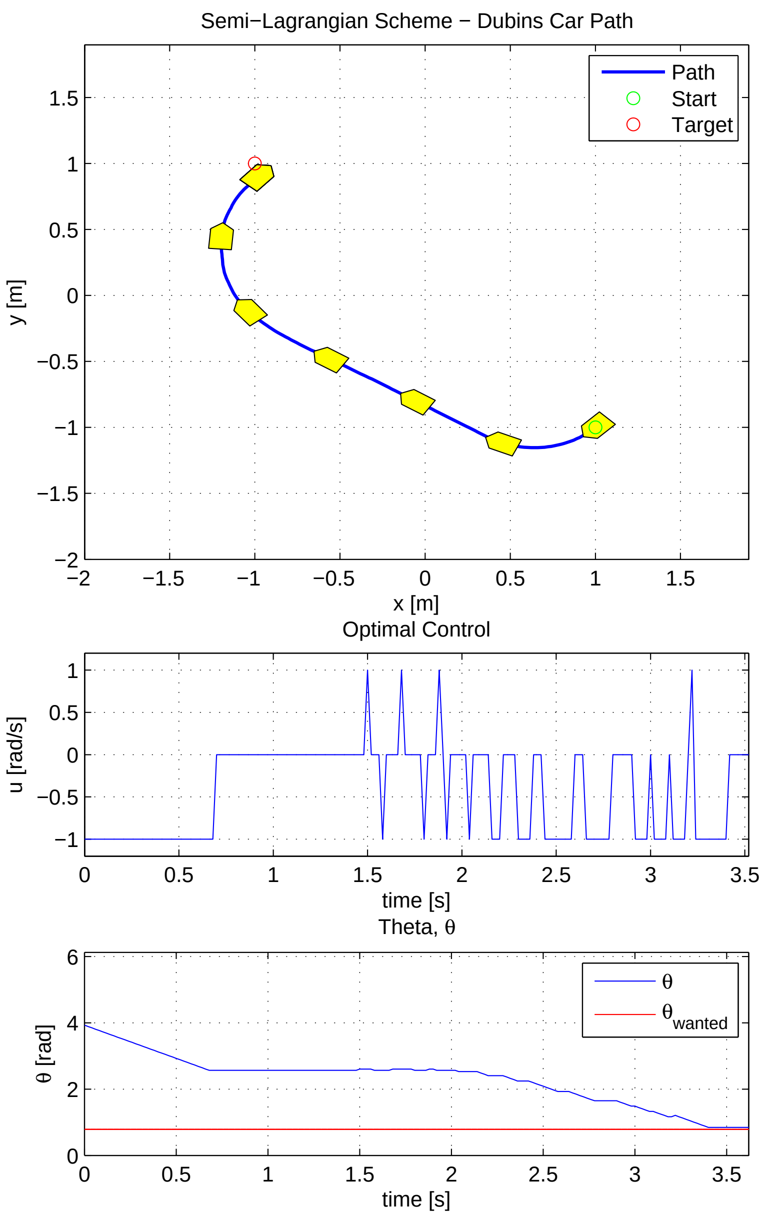
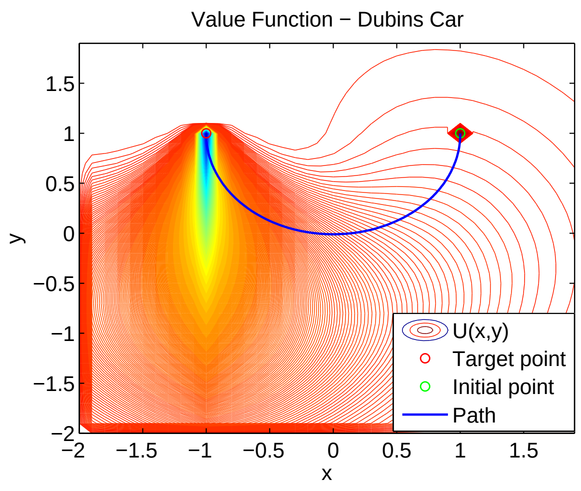
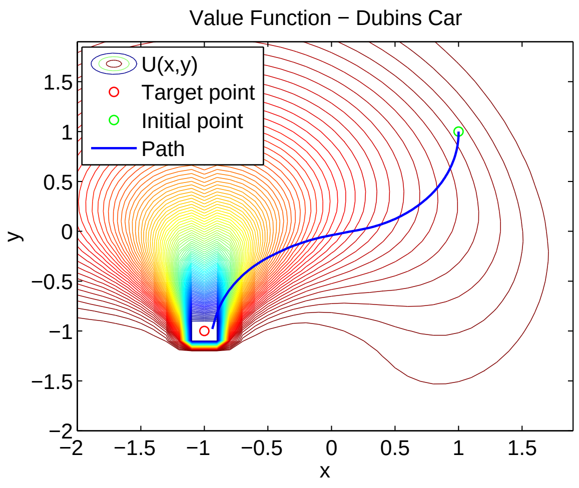


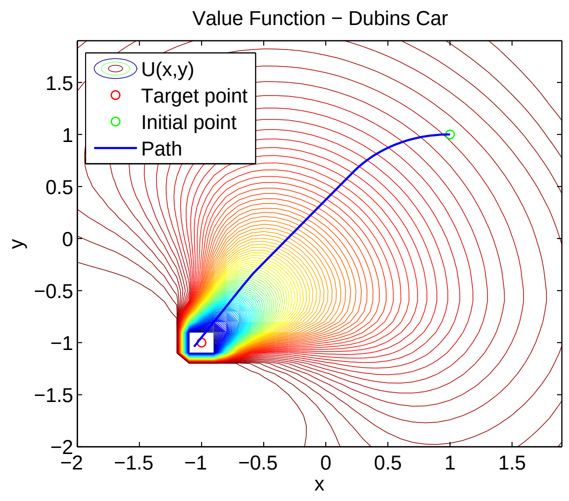
Figure 5.4: Dubins Car Using SL - Optimized Path from $(1, -1, \frac{5\pi}{4})$ to $(-1, 1, \frac{\pi}{4})$



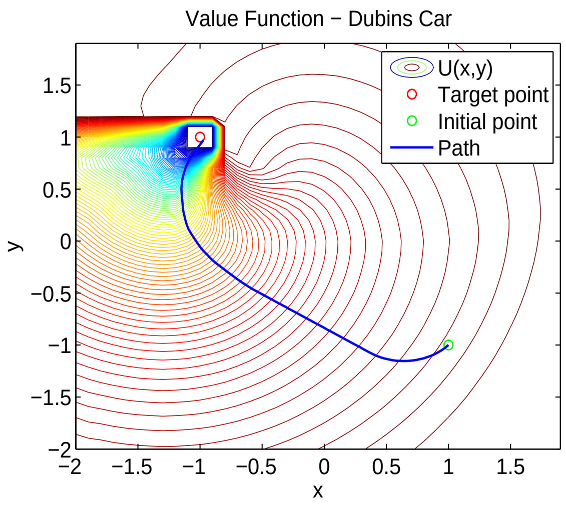
(a) For Optimized Path in Figure 5.1



(b) For Optimized Path in Figure 5.2

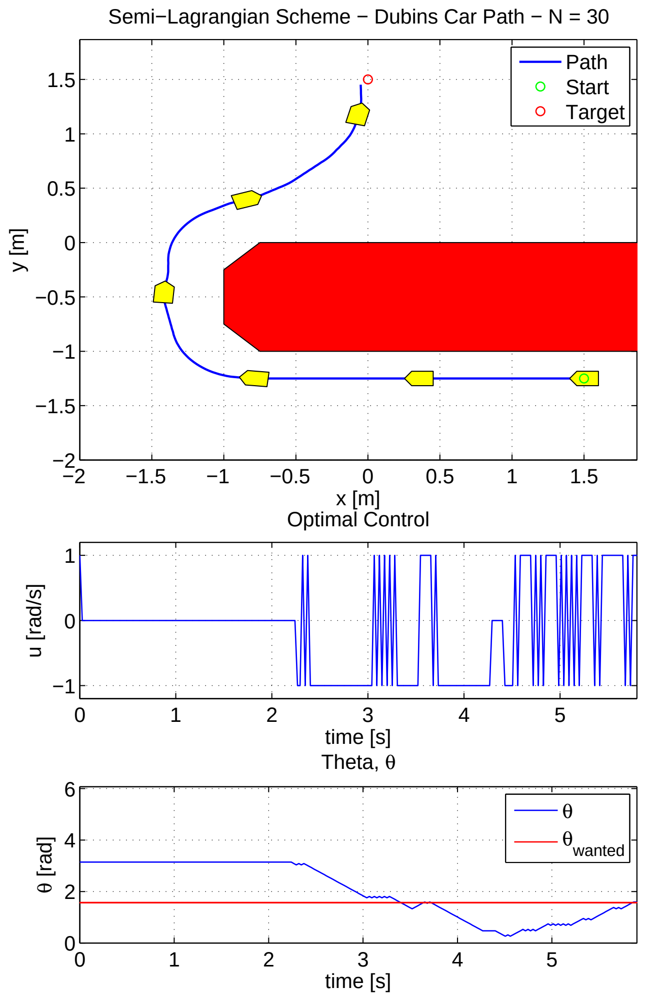


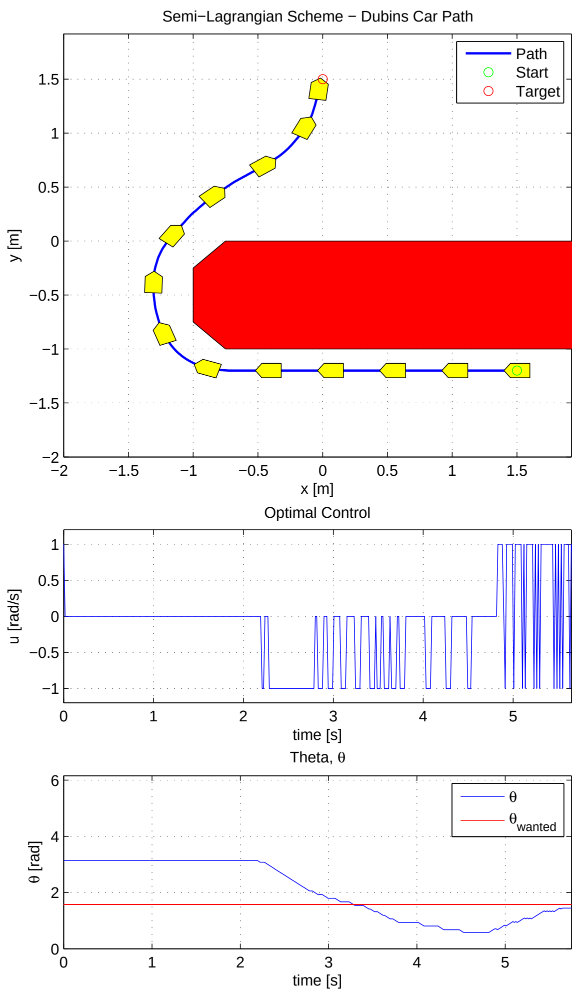
(c) For Optimized Path in Figure 5.3



(d) For Optimized Path in Figure 5.4

Figure 5.5: Value Functions for DU Car

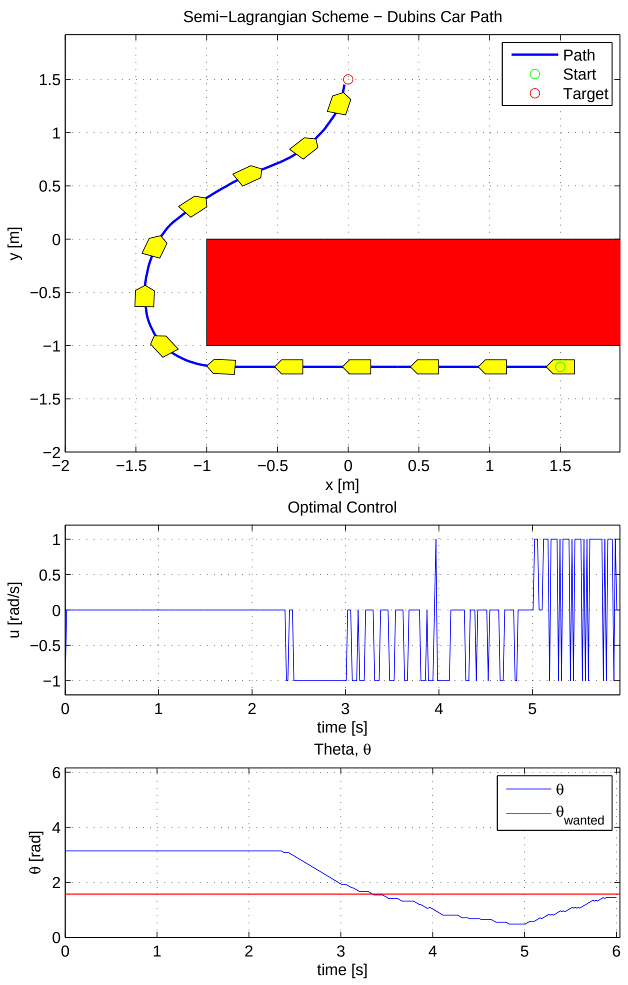




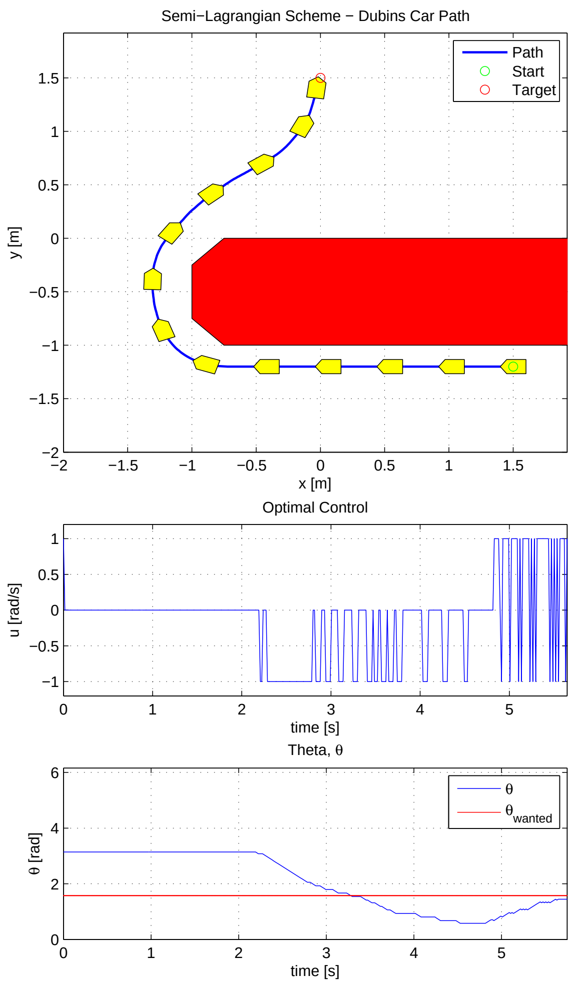
(a) Optimized Path for N = 30

(b) Optimized Path for N = 60

Figure 5.6: Dubins Car with Obstacles Using SL - Comparing N Values. Optimized path from $(1.5, -1.25, \pi)$ to $(0, 1.5, \frac{\pi}{2})$



(a) Optimized Path for Square Obstacle



(b) Optimized Path for Round Corner Obstacle

Figure 5.7: Dubins Car with Obstacles Using SL - Comparing Obstacle Shapes. Optimized path from $(1.5, -1.25, \pi)$ to $(0, 1.5, \frac{\pi}{2})$

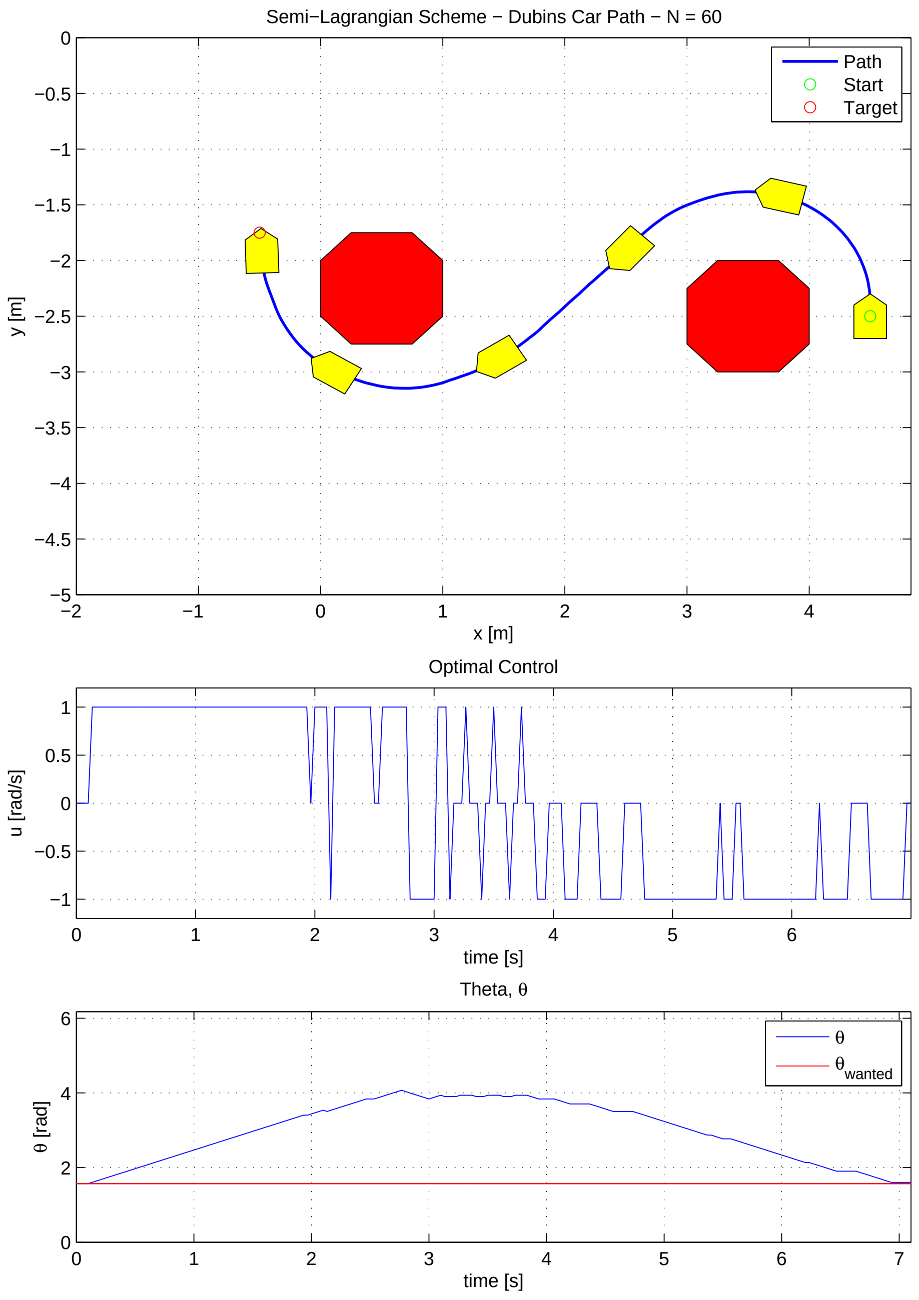
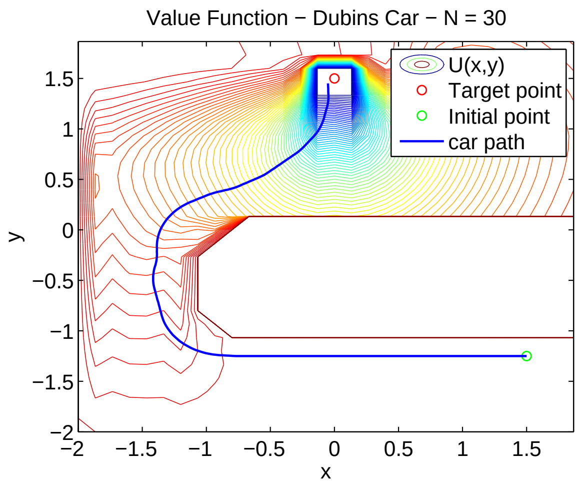
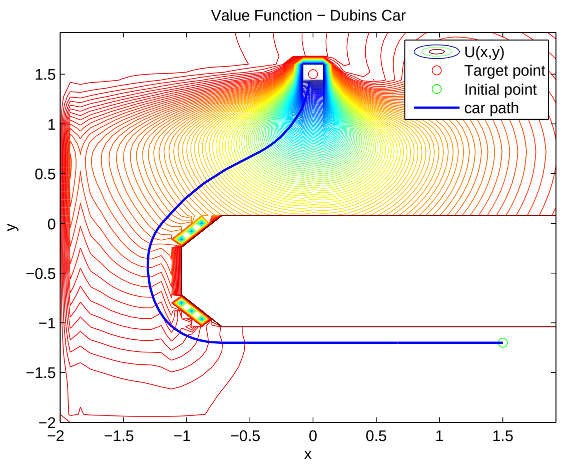


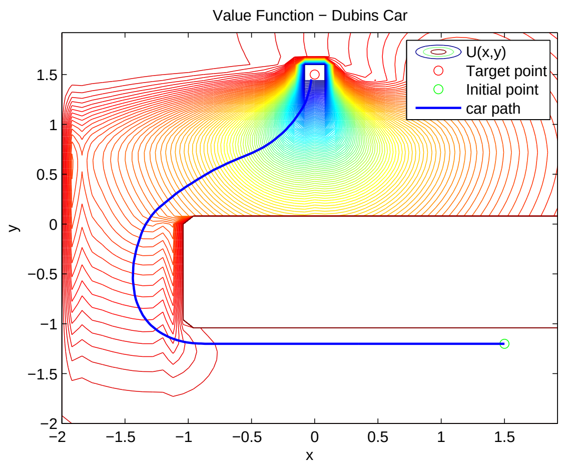
Figure 5.8: Dubins Car with Several Obstacles Using SL. Optimized path from $(4.5, -2.5, \frac{\pi}{2})$ to $(-0.5, -1.75, \frac{\pi}{2})$



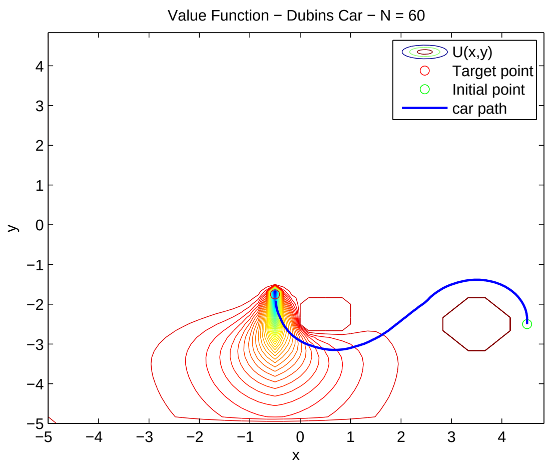
(a) For Optimized Path in Figure 5.6a



(b) For Optimized Path in Figure 5.6b



(c) For Optimized Path in Figure 5.7a



(d) For Optimized Path in Figure 5.8

Figure 5.9: Value Functions for DU Car with Obstacles

5.2 Reeds-Shepp Car Model

After looking at the shortest path for a Dubins car model, a nonholonomic mobile robot that can only move forwards, the search for the minimum time path is expanded to the Reeds-Shepp car. The Reeds-Shepp car is a car that can drive both forwards and backwards. In other words, the linear velocity constraint is $v(\cdot) \in \{-1, 1\}$. The model is described in more detail in Section 4.1. This section describes the calculations and results when solving the Hamilton-Jacobi-Bellman equation by using the SL scheme. Theory and details about HJB and the SL scheme are found in sections 3.1 and 3.3.

By using the RS model represented by (4.6) - (4.8) and the general HJB equation for a minimum time problem shown in (3.4), the following HJB equation has to be solved:

$$\begin{cases} V_t(\mathbf{x}, t) + \max\limits_{u_1 \in [-1,1], u_2 \in \{-1,1\}} \{V_x(\mathbf{x}, t) u_1 cos(\theta) + V_y(\mathbf{x}, t) u_1 sin(\theta) + V_\theta(\mathbf{x}, t) \dfrac{u_2}{\rho_{min}} + 1 = 0 \\ V(\mathbf{x}, T) = 0 \end{cases}$$

This is again solved using the steps described in Section 3.3.3. A detailed description of the steps in the implementation is also shown in Algorithm 1.

Table 5.3 shows the different test values used to demonstrate SL for the Reeds-Shepp car, and the corresponding figures of the optimized paths. The resulting trajectories are illustrated in figures 5.10 to 5.12. In these figures the green dot represents the starting point while the red dot represents the target point. The value functions for some of these tests are illustrated together with the optimized paths, in Figure 5.14.

Initial pose	Final pose	N	Domain Ω	h_x, h_y, h_θ	ρ_{min}	Figure
$(1, 1.5, \frac{3\pi}{2})$	$(0, -1, \frac{\pi}{2})$	40	$[-2, 2] \times [0, 2\pi)$	0.1, 0.1, 0.1571	0.5	Figure 5.10
(-1, 0, 0)	$(1, 0, \pi)$	40	$[-2, 2] \times [0, 2\pi)$	0.1, 0.1, 0.1571	0.5	Figure 5.11
$(0, -1, \pi)$	$(0, 1, \pi)$	40	$[-2, 2] \times [0, 2\pi)$	0.1, 0.1, 0.1571	0.5	Figure 5.12

Table 5.3: Test Values for Optimized Paths for RS Using SL Scheme

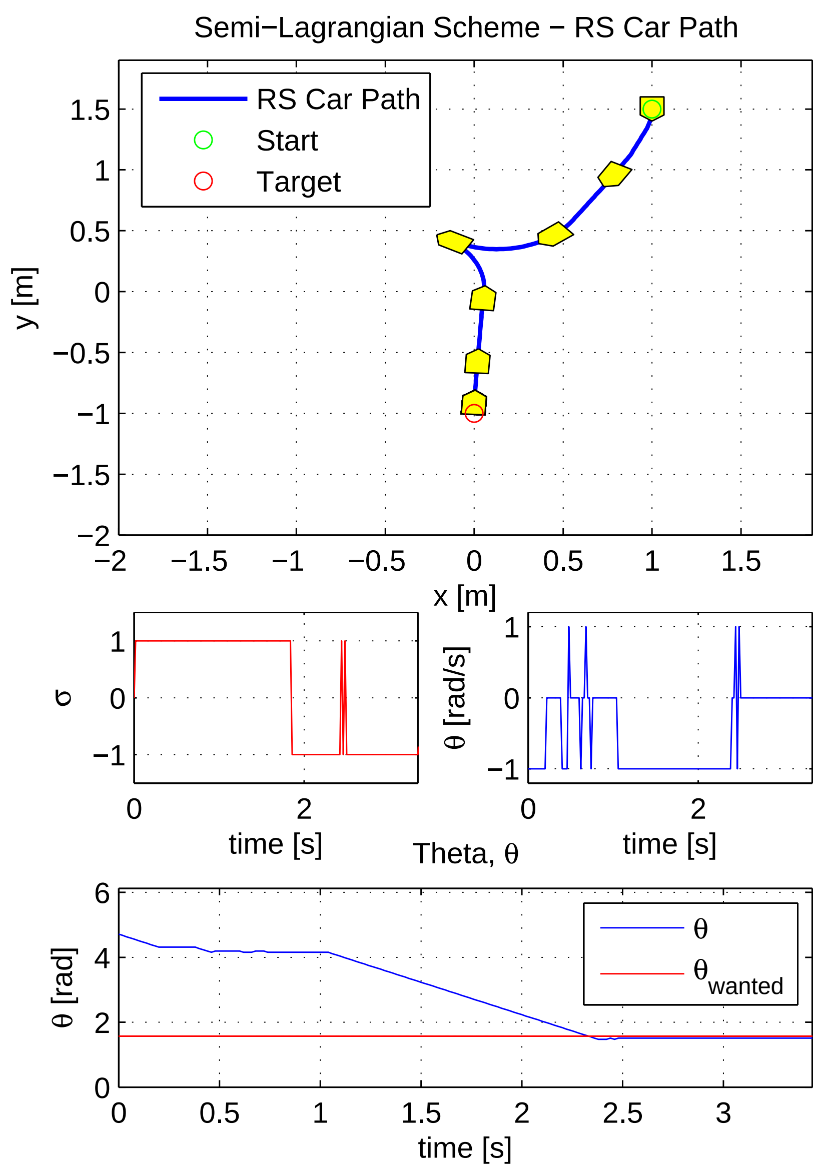


Figure 5.10: RS Car Using SL - Optimized Path from $(1, 1.5, \frac{3\pi}{2})$ to $(0, -1, \frac{\pi}{2})$

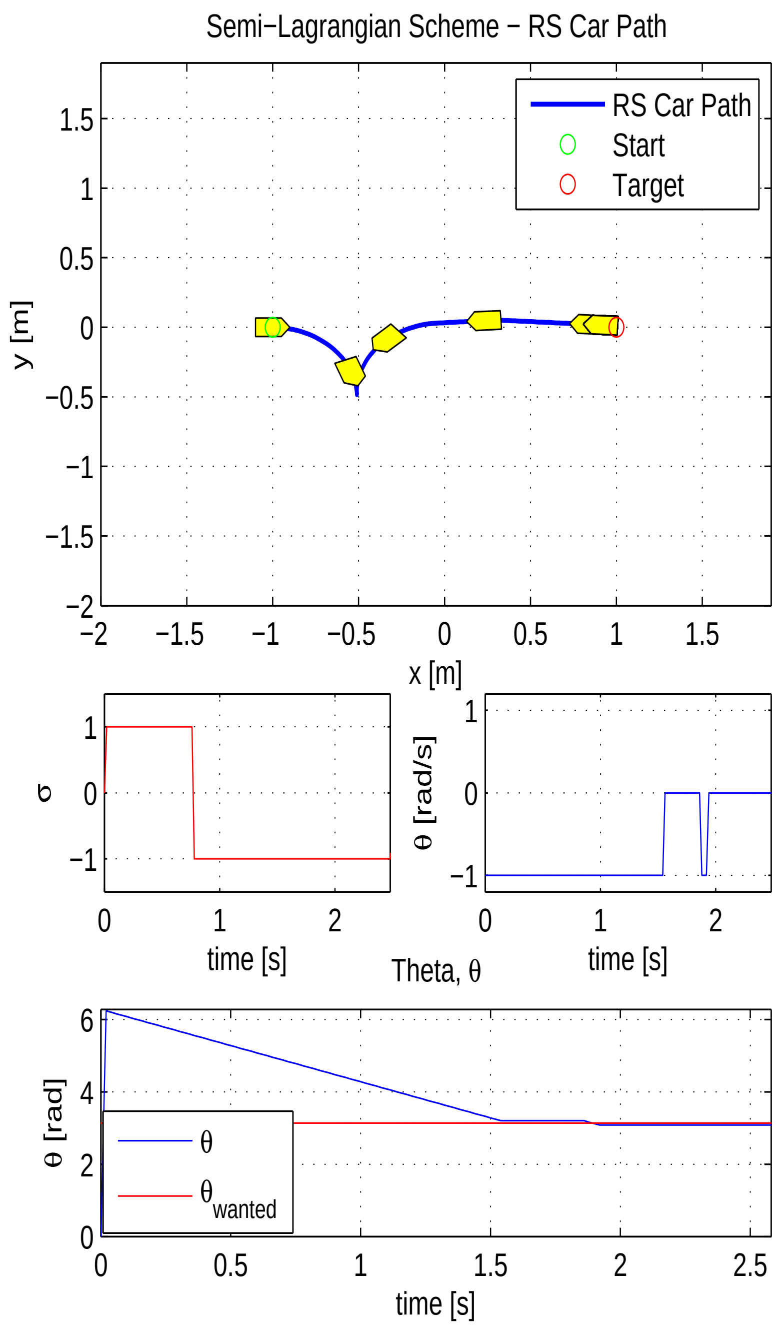


Figure 5.11: RS Car Using SL - Optimized Path from $(-1, 0, 0)$ to $(1, 0, \pi)$

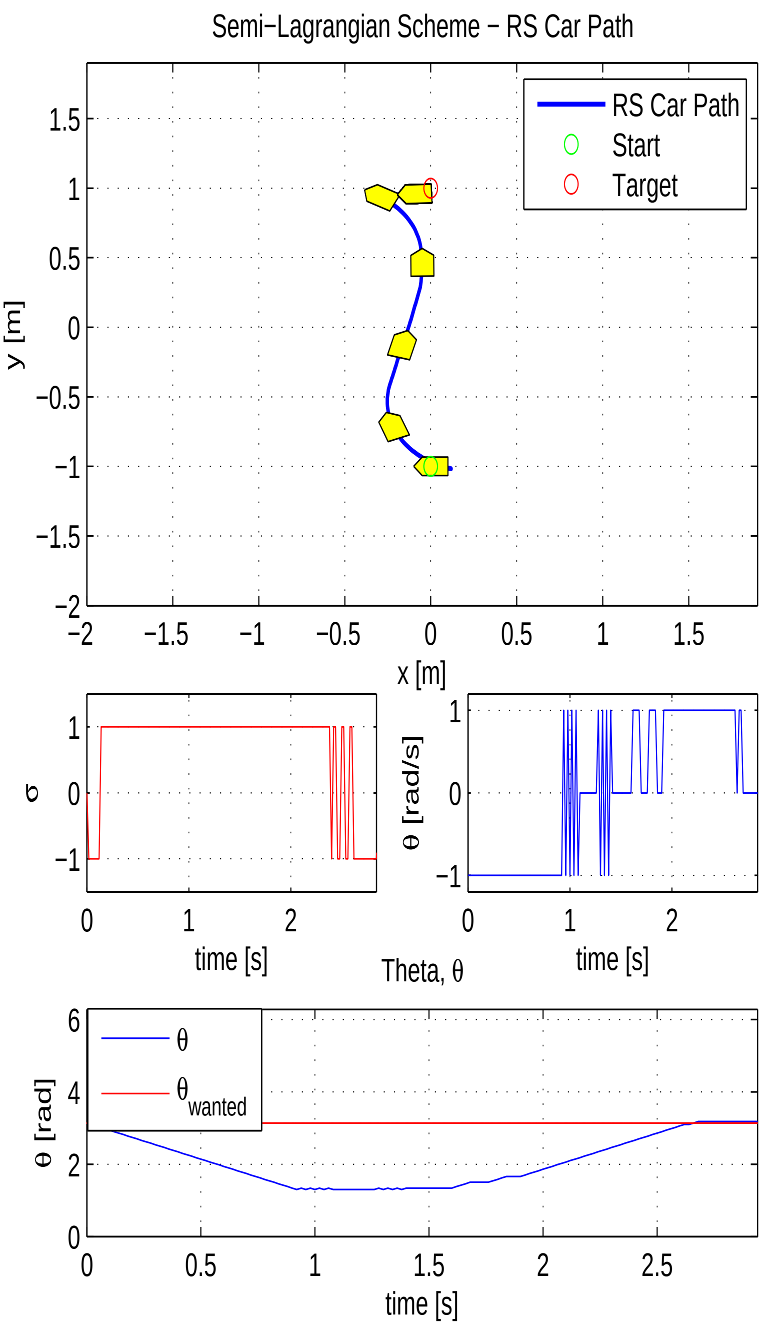


Figure 5.12: RS Car Using SL - Optimized Path from $(0, -1, \pi)$ to $(0, 1, \pi)$

Again, after implementing the minimum time paths in an obstacle-free environment, the implementation is expanded to include obstacles the car has to avoid. Obstacles were created and defined as a part of the subset $I_{out} = \{i \in I \setminus I_\Gamma : x_i + \Delta t f(x_i, u) \notin Q\}$ and extra code was added to make sure the value function $\mathbf{V}$ kept its initialized value at the nodes within the subset I_{out}.

Figure 5.13 shows one example of an optimized path for the RS car in an environment including obstacles that was created by this implementation. The corresponding test values and run conditions are shown in Table 5.4. This a typical "parallel parking" example where the car must manuever into a slot by driving both forwards and backwards. The value function for this test is illustrated in Figure 5.14. Again, oscillation in the control input appears. In addition to switching between left (u_1 = 1) and right turn (u_1 = -1) instead of going straight forward ($u_1 = 0$), the vehicle alternates several times between driving forward and backward while taking many small turns instead of taking one big turn.

Initial pose	Final pose	N	Domain Ω	h_x, h_y, h_θ	ρ_{min}	Figure
$(1, 0.25, \pi)$	$(-0.75, -1, \pi)$	40	$[-2, 2] \times [0, 2\pi)$	0.1, 0.1, 0.1571	0.5	5.13

Table 5.4: Test Values for Optimized Paths for RS with Obstacles Using SL Scheme

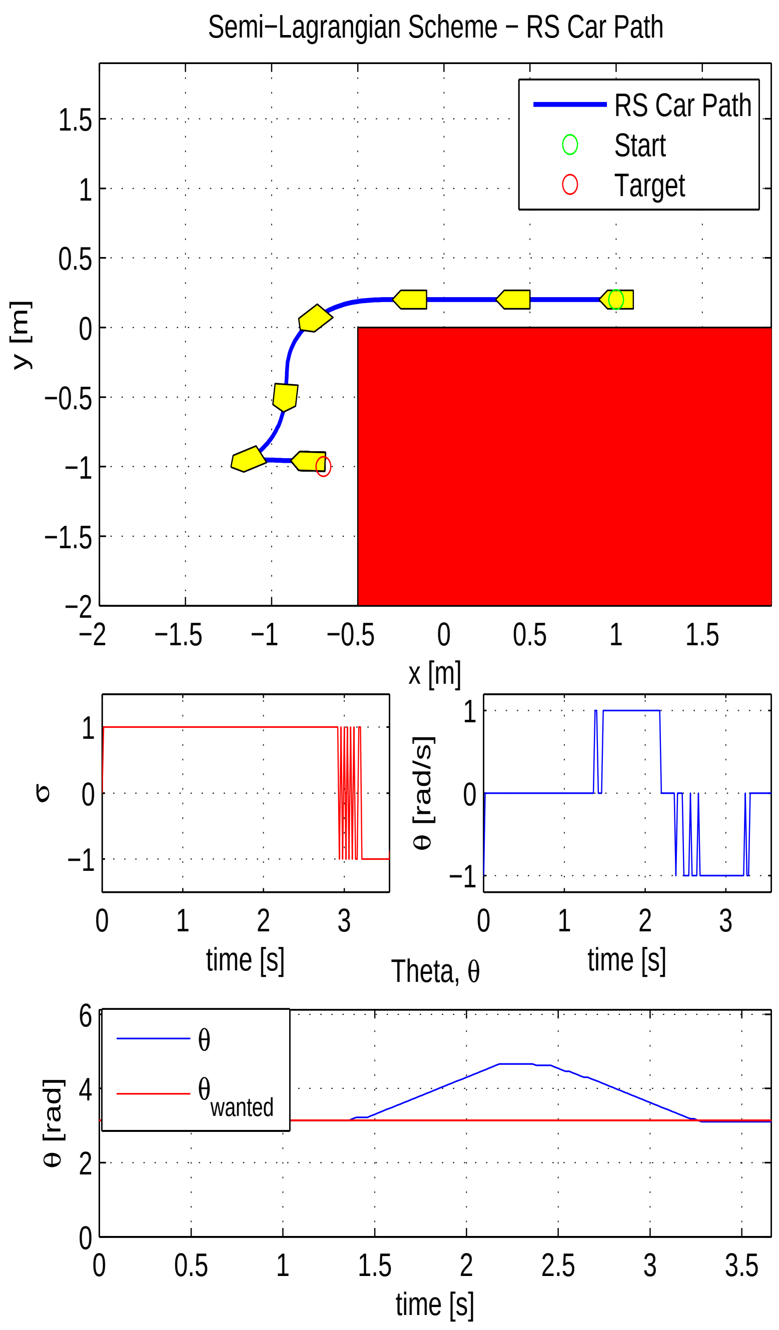
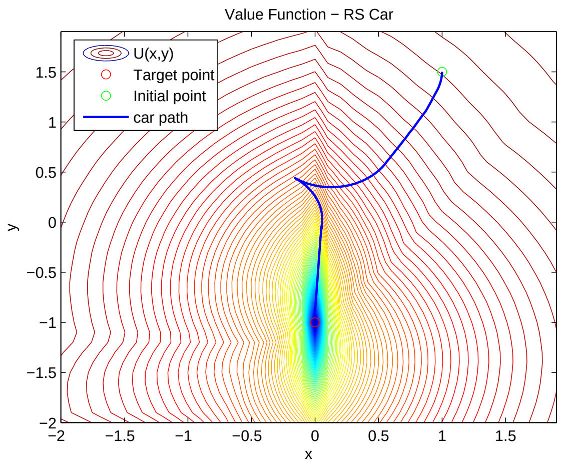
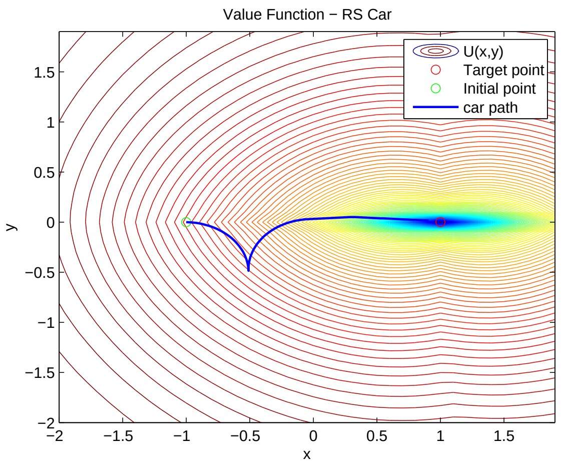


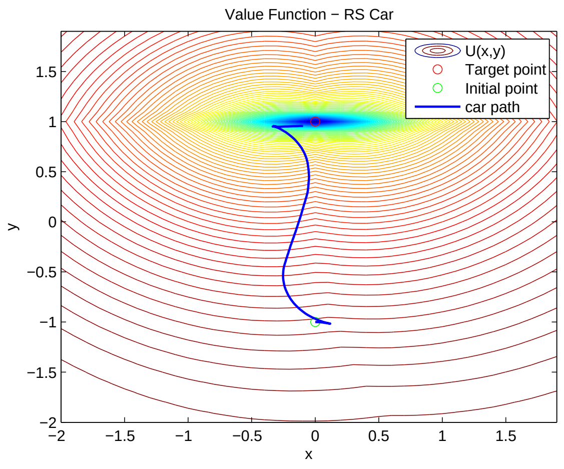
Figure 5.13: RS Car with Obstacles Using SL. Optimized path from $(1, 0.25, \pi)$ to $(-0.75, -1, \pi)$ with obstacle



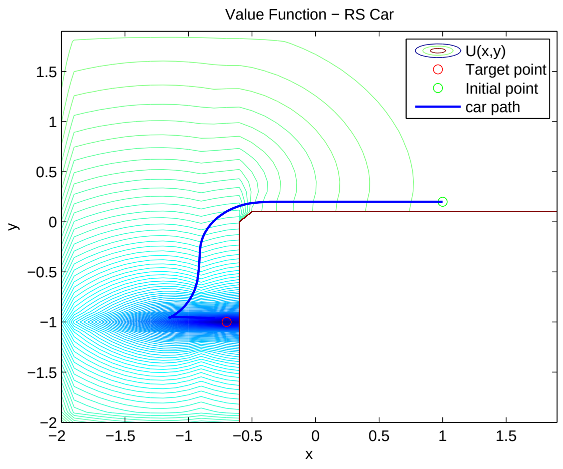
(a) For Optimized Path in Figure 5.10



(b) For Optimized Path in Figure 5.11



(c) For Optimized Path in Figure 5.12



(d) For Optimized Path in Figure 5.13

Figure 5.14: Value Functions for RS Car

5.3 Boat Model

This section describes the calculations and results when solving the Hamilton-Jacobi-Bellman equation by using the SL scheme to produce the shortest path for a simplified boat model. The theory used to conduct the implementation, being HJB, SL scheme and the simplified boat model, is found in sections 3.1, 3.3 and 4.2.

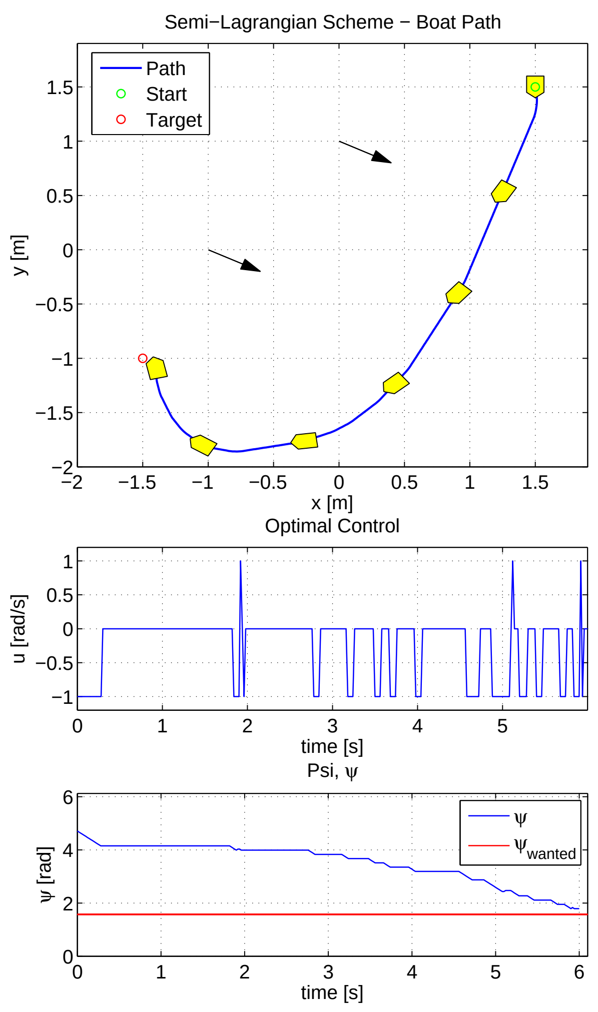
The model first tested included both surge and sway velocities and is described by (4.9) - (4.13). The SL scheme then had to run and calculate a value function while considering all 5 dimensions (x, y, ψ, u and v). It was quickly discovered that the SL scheme did not converge quickly enough with five dimensions, so the boat model was simplified by assuming that the speed in sway direction is so minimal that it can be neglected in the model. Instead, ocean current, another important factor when modelling a boat, was introduced. The model used for the implementation is described by (4.17) - (4.19).

Since the ocean current is unknown, it is not included in the SL scheme calculation of the value function. The SL scheme will therefore calculate a value function that assumes no presence of ocean current. When using this value function to calculate the optimized control action, the resulting optimized path strays away from the optimal path because it does not account for or counteract the ocean current. The current will therefore, in general, push the boat away from the path. The boat is, however, with a slight detour and a somewhat uneven path, able to get to the target. Even though the value function is calculated based on no current, it will still draw the vehicle towards the target. The control action is calculated at each timestep based on the closest current nodes and the value function at these nodes. This causes the boat to steer towards the target, although having a slight detour because the value function is based on the wrong ocean current information.

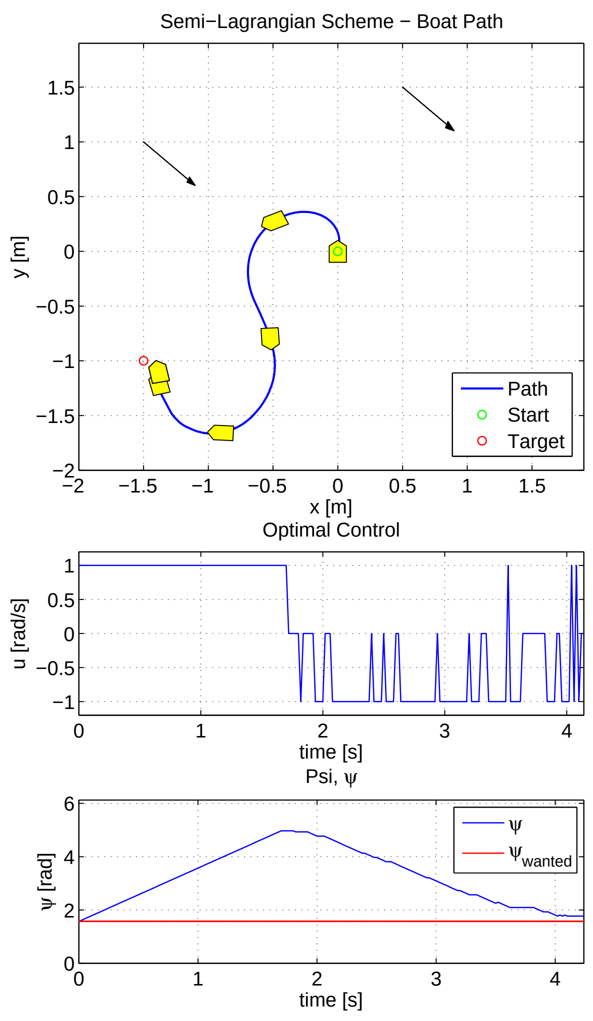
Figure 5.15 shows two examples of optimized paths calculated for this boat model using the SL scheme. The corresponding test values and run conditions are shown in Table 5.5.

Initial pose	Final pose	N	Domain Ω	$h_x/h_y, h_\theta$	ρ_{min}	V_x, V_y
$(1.5, 1.5, -\frac{\pi}{2})$	$(-1.5, -1, \frac{\pi}{2})$	40	$[-2, 2] \times [0, 2\pi)$	0.1, 0.1571	0.5	0.2 m/s, -0.1 m/s
$(0, 0, \frac{\pi}{2})$	$(-1.5, -1, \frac{\pi}{2})$	40	$[-2, 2] \times [0, 2\pi)$	0.1, 0.1571	0.5	0.2 m/s, -0.2 m/s

Table 5.5: Test Values for Optimized Paths for Boat Model Using SL Scheme



(a) Path from $(1.5, 1.5, -\frac{\pi}{2})$ to $(-1.5, -1, \frac{\pi}{2})$



(b) Path from $(0, 0, \frac{\pi}{2})$ to $(-1.5, -1, \frac{\pi}{2})$

Figure 5.15: Boat with Ocean Current Using SL

Chapter 6

Pseudospectral Implementation and Results

In this chapter the design, implementation and results when using the MATLAB application package DIDO and the pseudospectral method to solve the shortest path for the different models, as described in Chapter 4, are presented.

6.1 Dubins Car Model

This section describes the calculations and results when solving the optimal control problem by using the MATLAB application DIDO and the PS method to produce the minimum time path for a Dubins car. The theory used to conduct the implementation is found in sections 3.4 and 4.1.

The model used is represented by (4.3) - (4.5). By defining the boundaries on the state and controls of the system and by setting up the code structure in a manner similar to the example code shown in [44] and [26], DIDO was ready to calculate the optimized path.

A selection of the minimum time paths given by this implementation is shown in figures 6.1 to 6.4. The corresponding test values and run conditions are shown in Table 6.1. All paths are very close to the optimal paths described by Dubins paths in Section 4.1.1. The only deviation is that there are some slight oscillations in the control action.

Initial pose	Final pose	N	Domain Ω	ρ_{min}	Figure
$(1, 1, \frac{3\pi}{2})$	$(-1, -1, \frac{\pi}{2})$	60	$[-2, 2] \times [0, 2\pi)$	0.5	Figure 6.1
$(0, 1.5, 0)$	$(0, -1.5, 0)$	60	$[-2, 2] \times [0, 2\pi)$	0.5	Figure 6.2
$(0, 0, 0)$	$(-0.5, 0, \pi)$	60	$[-2, 2] \times [0, 2\pi)$	0.5	Figure 6.3
$(1, 1, \frac{3\pi}{2})$	$(-1, -1, \frac{3\pi}{2})$	60	$[-2, 2] \times [0, 2\pi)$	0.5	Figure 6.4

Table 6.1: Test Values for Optimized Paths for Dubins Car Using PS Method

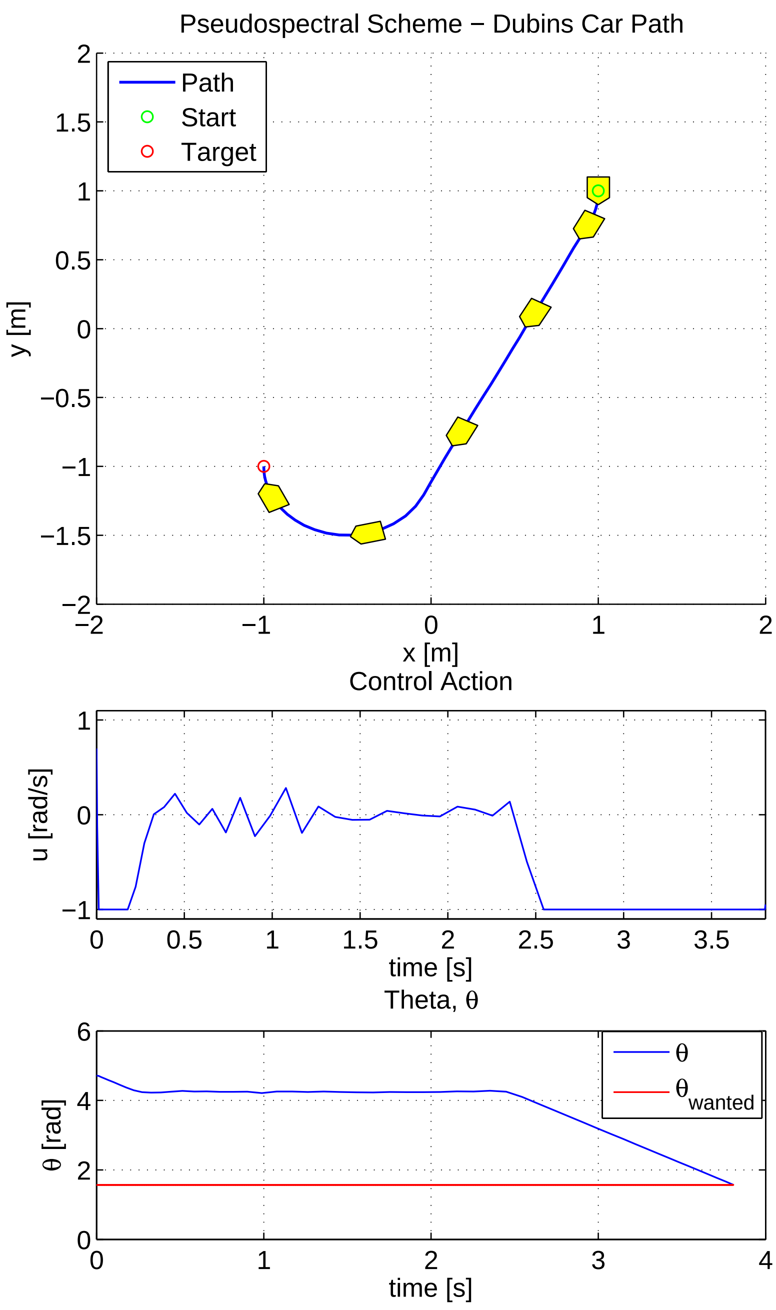


Figure 6.1: Dubins Car Using PS - Optimized Path from $(1, 1, \frac{3\pi}{2})$ to $(-1, -1, \frac{\pi}{2})$

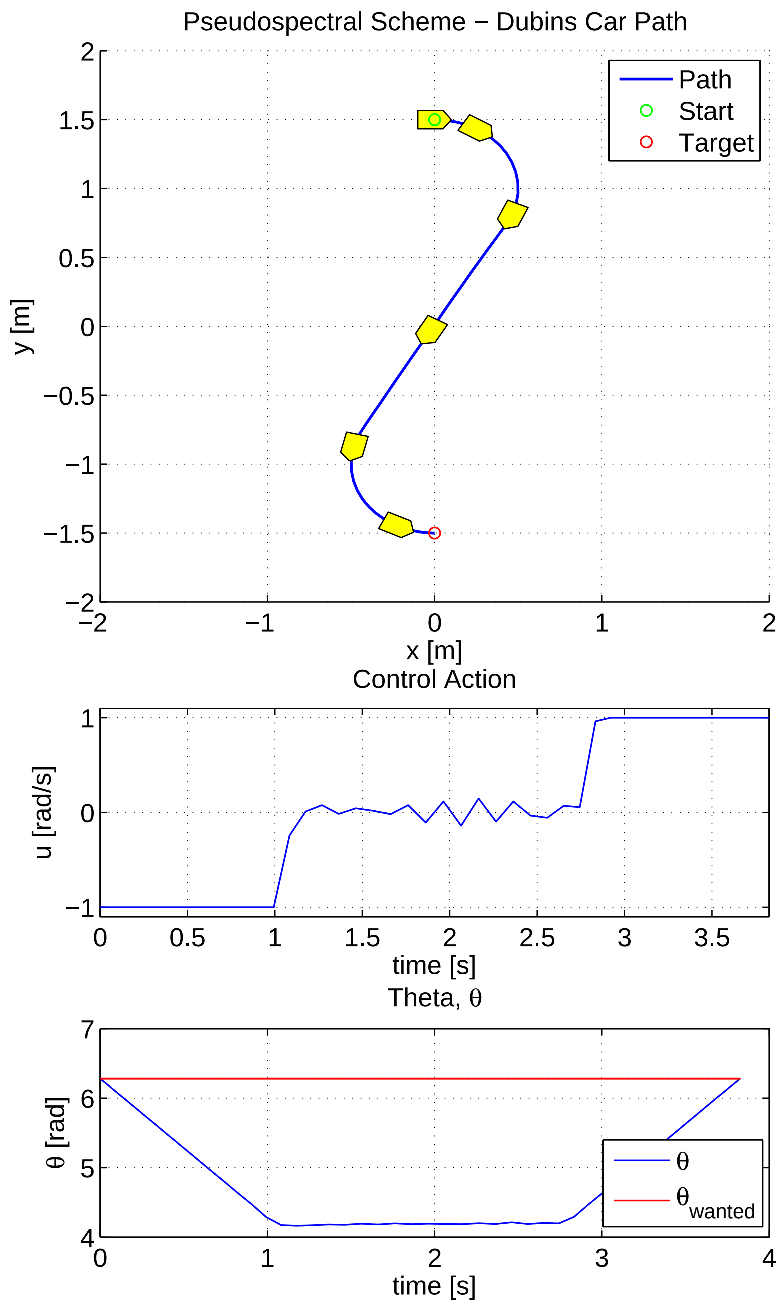


Figure 6.2: Dubins Car Using PS - Optimized Path from $(0, 1.5, 0)$ to $(0, -1.5, 0)$

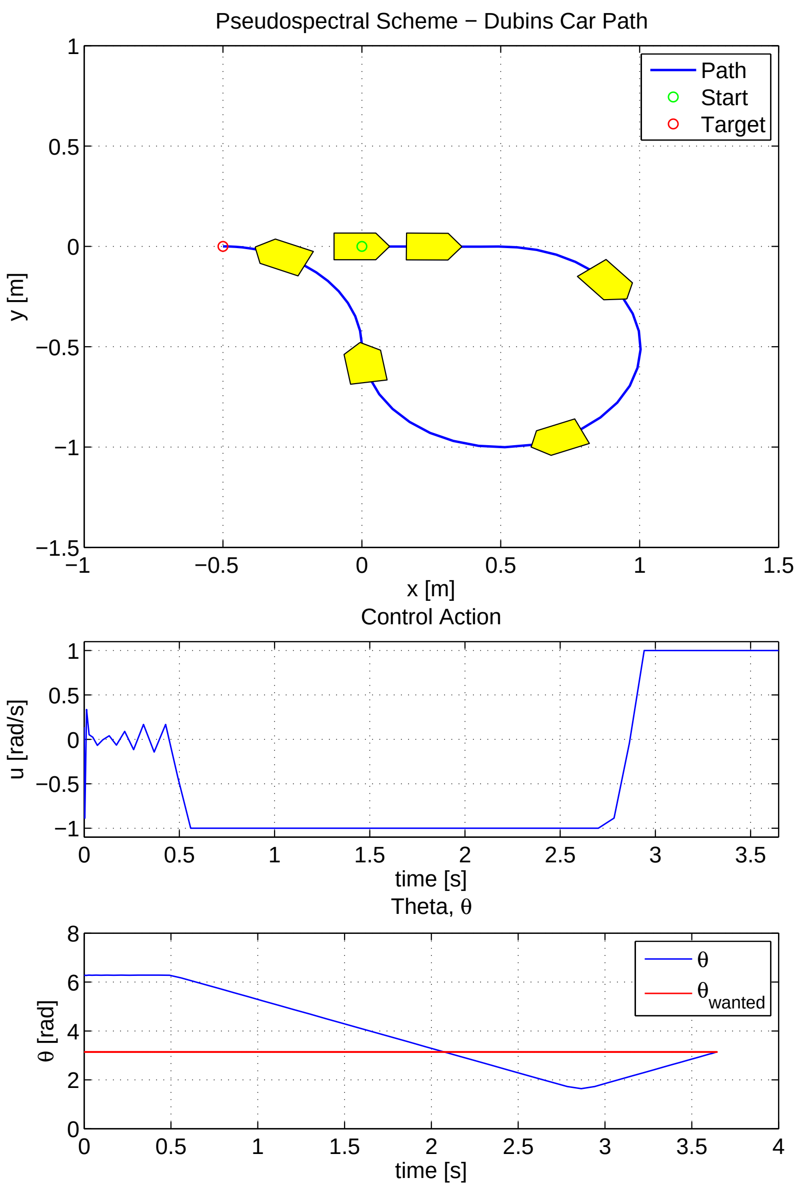


Figure 6.3: Dubins Car Using PS - Optimized Path from $(0, 0, 0)$ to $(-0.5, 0, \pi)$

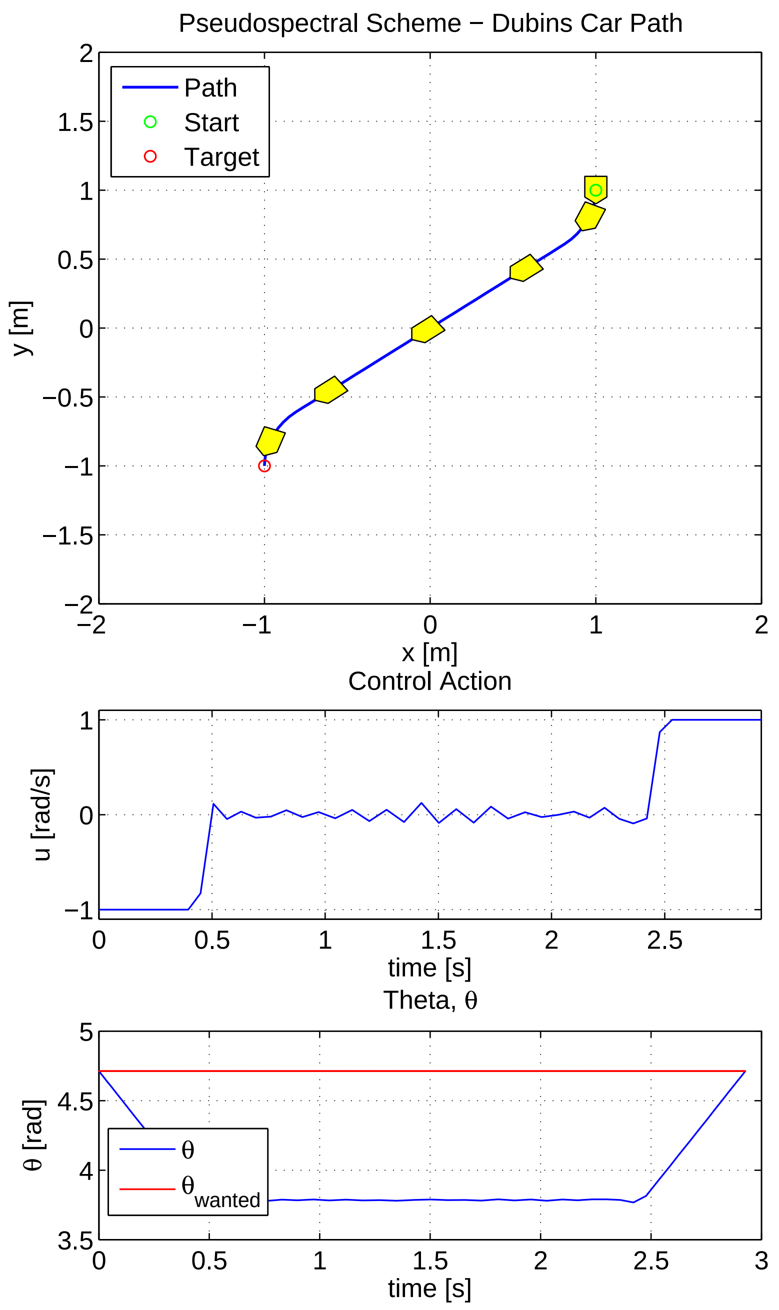


Figure 6.4: Dubins Car Using PS - Optimized Path from $(1, 1, \frac{3\pi}{2})$ to $(-1, -1, \frac{3\pi}{2})$

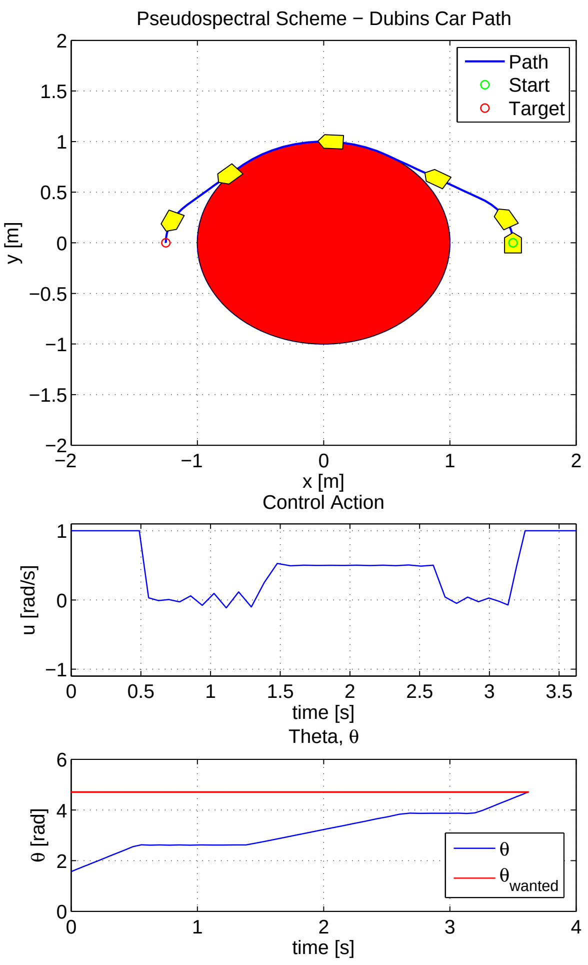
After implementing the minimum time paths for the Dubins car in an obstacle-free environment, the implementation was expanded to include obstacles the car had to avoid. To achieve this, obstacles had to be created and additional code had to be added to include path constraints for these obstacles. Defining the path constrains means defining a function h, with upper and lower boundaries. These additional conditions, which remove some positions from the available "legal" positions in the optimization problem, must be held at all node points.

Tests were run with both a single obstacle and two obstacles. A comparison of different obstacle clearances was also done to illustrate this difference. With no obstacle clearance the optimized path calculated by DIDO often touches the object. In general, this will not be possible because the vehicle itself is wider than a singe spot and needs this extra space to maneuver around the obstacle. In many situations it is also necessary to keep some distance to obstacles for safety purposes and in order to have extra margins for the vessel's ability to navigate correctly.

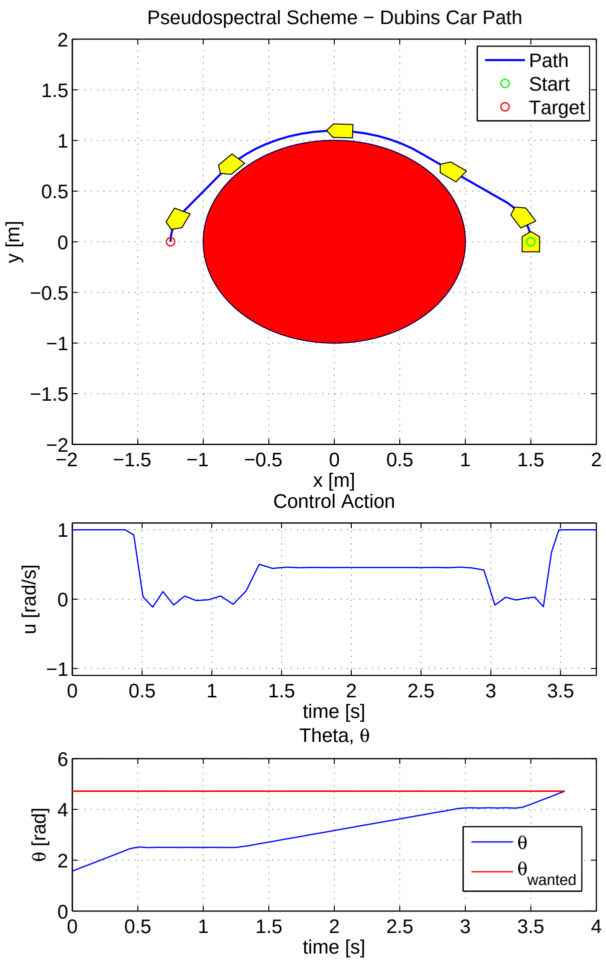
The test runs for a Dubins car with obstacles are shown in figures 6.5 and 6.6. The corresponding test values and run conditions are shown in Table 6.2. When obstacles are in the way of the shortest path, it is seen that the paths calculated by PS hug the border of the obstacles. It is therefore even more important that a safety margin is included. Since the optimized paths go so close to the target, these paths are shorter then the paths given by SL.

Initial pose	Final pose	N	Domain Ω	ρ_{min}	Obstacle Clearance	Figure
$(1.5, 0, \frac{\pi}{2})$	$(-1.2, 0, \frac{3\pi}{2})$	60	$[-2, 2] \times [0, 2\pi)$	0.5	0	Figure 6.5a
$(1.5, 0, \frac{\pi}{2})$	$(-1.2, 0, \frac{3\pi}{2})$	60	$[-2, 2] \times [0, 2\pi)$	0.5	0.2	Figure 6.5b
$(3, -4, \pi)$	$(0, 1.5, \pi)$	60	$[-5, 5] \times [0, 2\pi)$	1	0	Figure 6.6

Table 6.2: Test Values for Optimized Paths for Dubins with Obstacles Using PS Method



(a) Optimized Paths Regular Obstacle Area



(b) Optimized Paths Expanded Obstacle Area

Figure 6.5: Dubins Car with Obstacles Using PS - Comparing Obstacle Clearance. Optimized path from $(1.5, 0, \frac{\pi}{2})$ to $(-1.2, 0, \frac{3\pi}{2})$

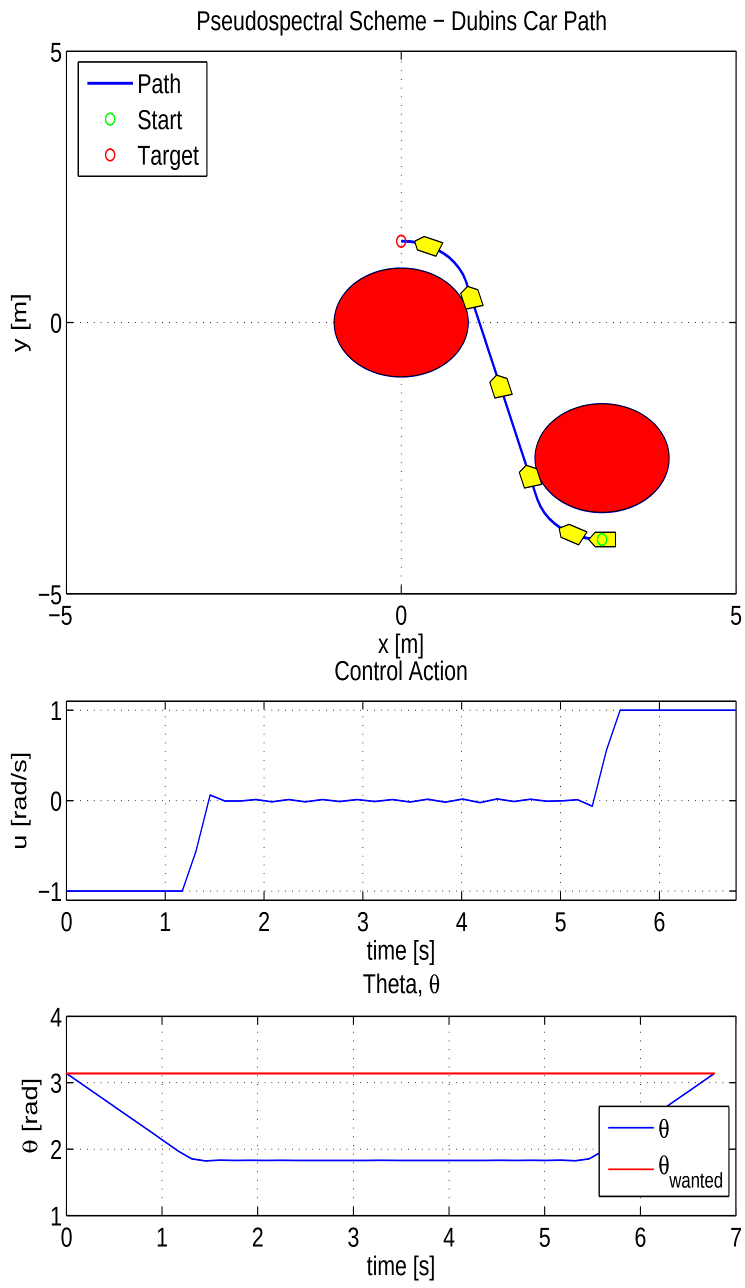


Figure 6.6: Dubins Car with Several Obstacles Using PS. Optimized path from $(3, -4, \pi)$ to $(0, 1.5, \pi)$

6.2 Reeds-Shepp Car Model

After looking at the shortest path for a Dubins car model, a nonholonomic mobile robot that can only move forwards, the search for the minimum time path was expanded to the Reeds-Shepp car. The Reeds-Shepp car is a car that can drive both forwards and backwards. In other words, the linear velocity constraint is $v(\cdot) \in \{-1, 1\}$. The model is described in more detail in Section 4.1.

This section describes the calculations and results when solving the optimal control problem by using the MATLAB application DIDO and the PS method to produce the minimum time path for an RS car. The theory used to conduct the implementation is found in sections 3.4 and 4.1.

The RS model used in this section is represented by (4.6) - (4.8). By redefining the boundaries on the state and controls of the system and setting up the code structure for an RS car, DIDO was ready to calculate the optimized path.

A selection of the minimum time paths given by this implementation is shown in figures 6.7 to 6.9. The corresponding test values and run conditions are shown in Table 6.3. Again, all paths are very close to the optimal paths described in Section 4.1.2, the only deviation being the small oscillations in the control action.

Initial pose	Final pose	N	Domain Ω	ρ_{min}	Figure
$(2, 3, \pi)$	$(0, 0, 0)$	60	$[-5, 5] \times [0, 2\pi)$	1	Figure 6.7
$(-1, 1, \frac{\pi}{2})$	$(1, -1, \pi)$	60	$[-2, 2] \times [0, 2\pi)$	0.5	Figure 6.8
$(-1, 0, 0)$	$(1, 0, \pi)$	60	$[-2, 2] \times [0, 2\pi)$	0.5	Figure 6.9

Table 6.3: Test Values for Optimized Paths for RS Using PS Method

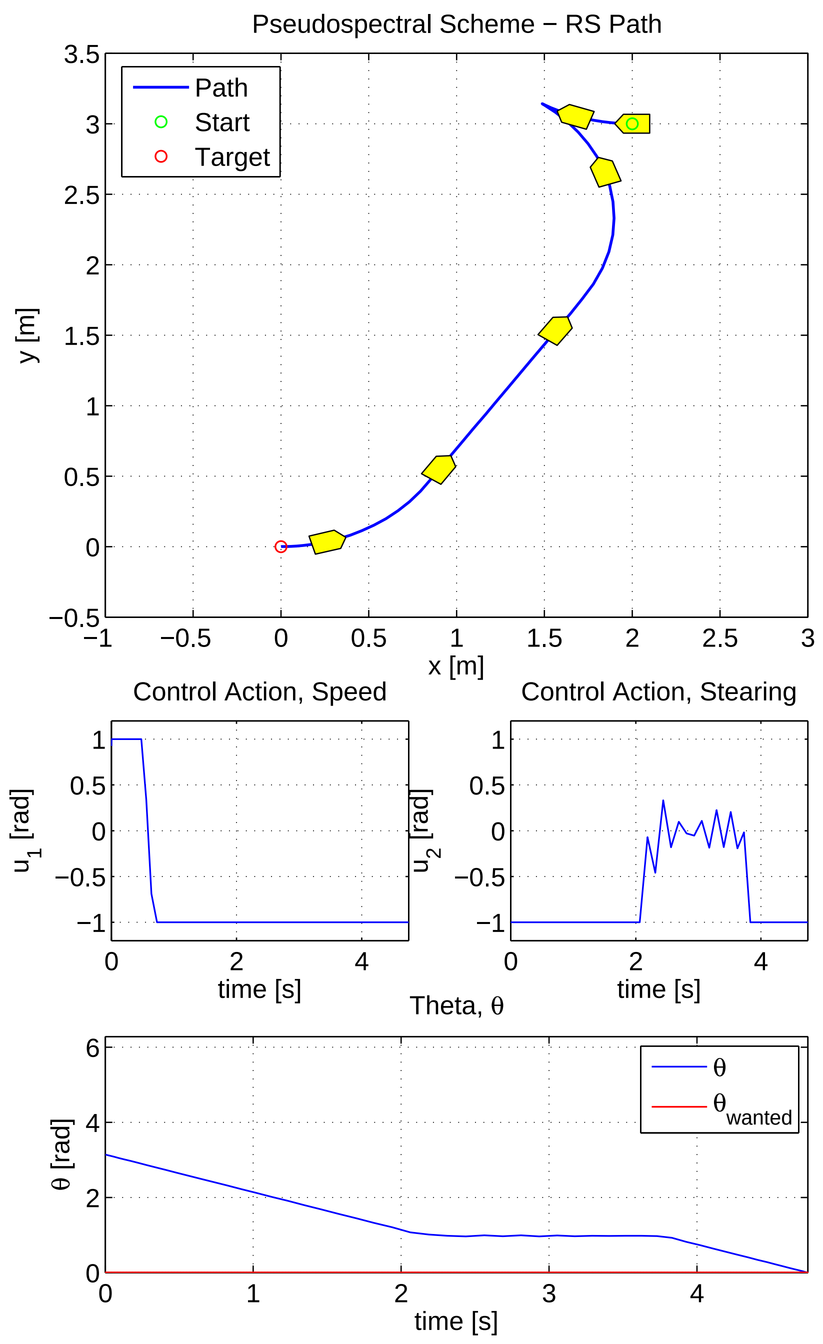


Figure 6.7: RS Car Using PS - Optimized Path from $(2, 3, \pi)$ to $(0, 0, 0)$

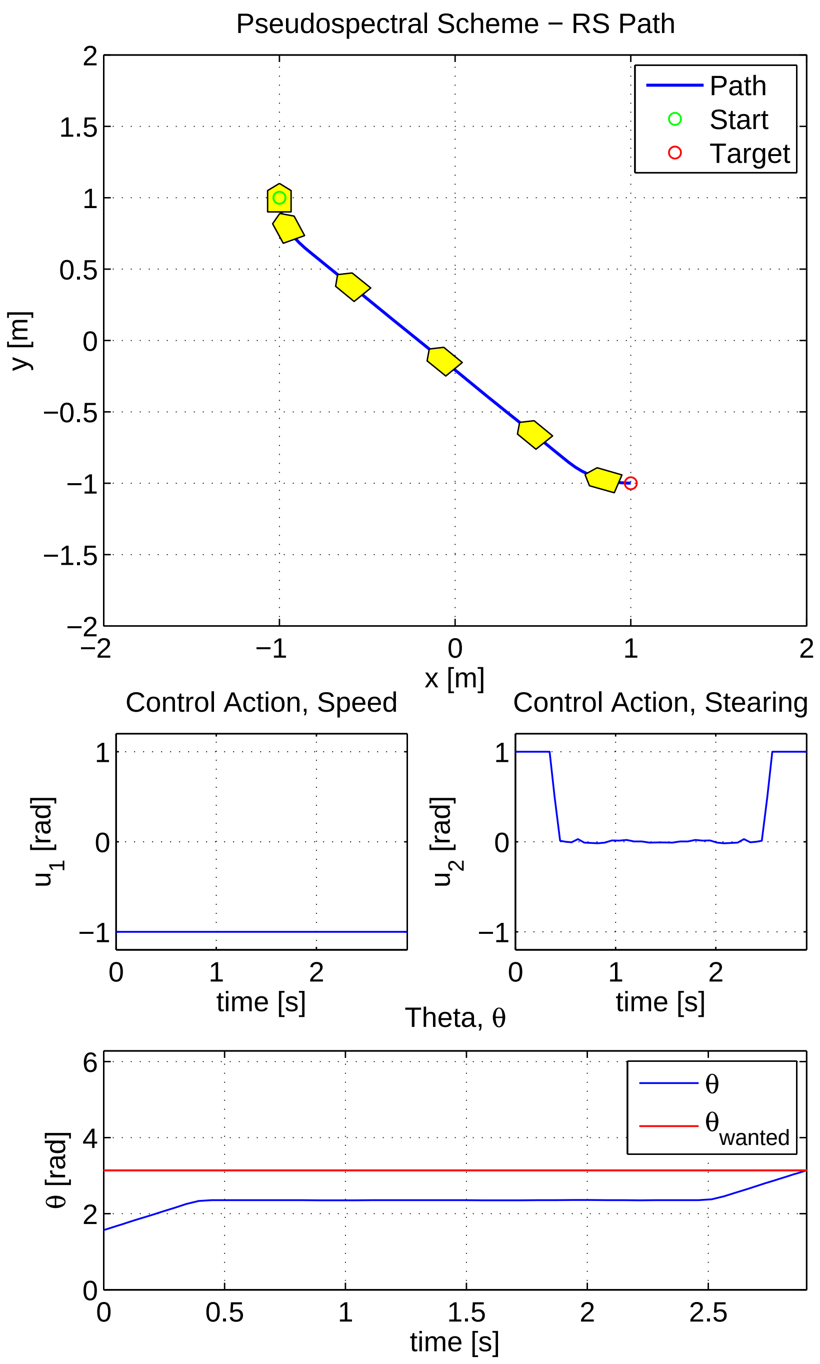


Figure 6.8: RS Car Using PS - Optimized Path from $(-1, 1, \frac{\pi}{2})$ to $(1, -1, \pi)$

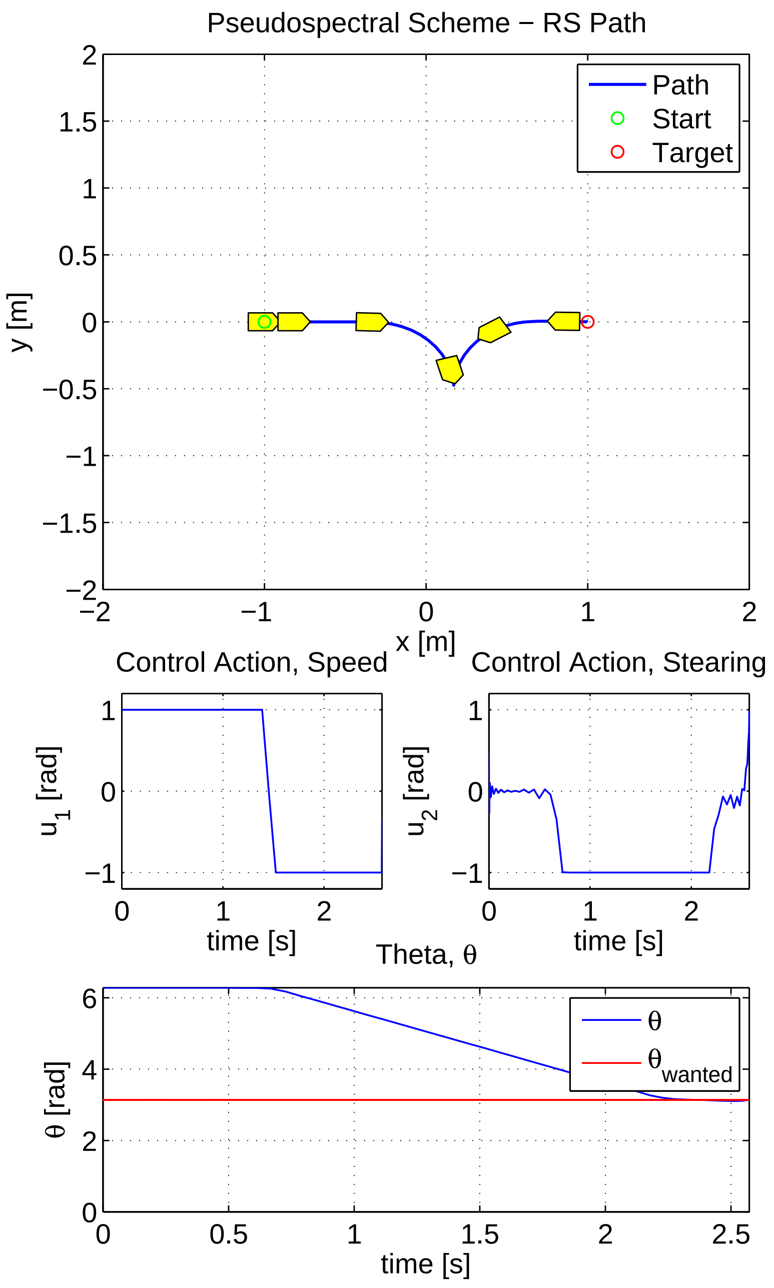


Figure 6.9: RS Car Using PS - Optimized Path from $(-1, 0, 0)$ to $(1, 0, \pi)$

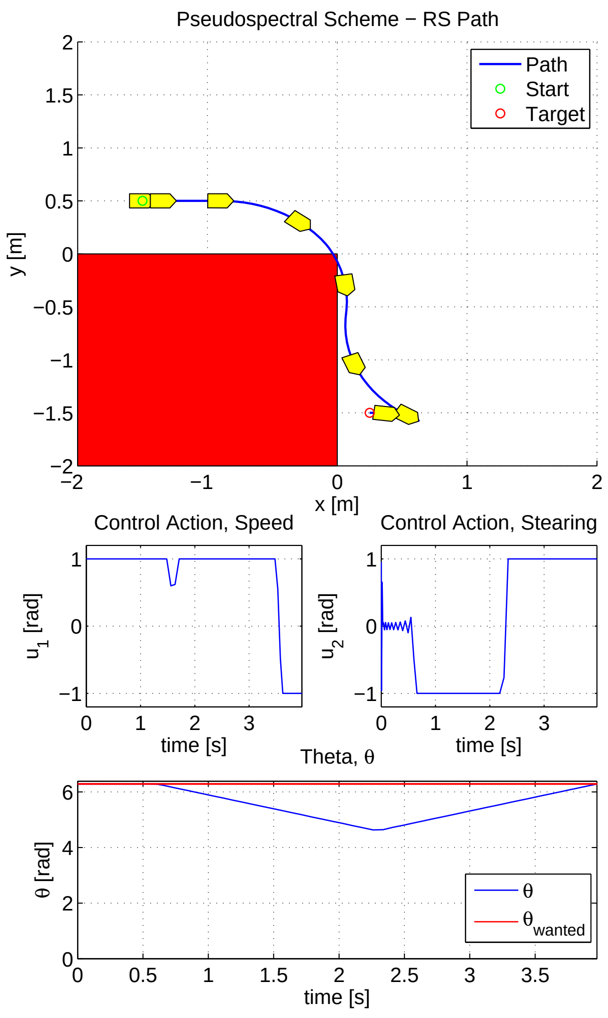
After implementing the minimum time paths for the RS car in an obstacle free-environment, the implementation was expanded to include obstacles the car had to avoid. To achieve this, obstacles had to be created and again additional code had to be added to include path constraints for these obstacles. These additional constraints must be held at all node points.

Tests were run for a "parallel parking" example. A comparison of different number of nodes was done to illustrate the effects on the path calculated. As seen in Figure 6.11, a higher number of nodes does not, for the PS method, automatically lead to a shorter or more accurate path. A comparison of different obstacle clearances was also done to illustrate the effects on the path calculated. In general, having a path with no obstacle clearance is not be possible because the vehicle itself is wider than a singe spot and needs this extra space to maneuver around the obstacle. In many situations it is also necessary to keep some distance to obstacles for safety purposes. As for the PS method it is seen in Figure 6.10 that the test with no obstacle clearance does not provide a feasible path at all. DIDO and PS discretize the trajectory with respect to time. The method ensures that at each node, optimality and satisfaction of the constraints are always ensured. The path that is propagated between the nodes still does not always satisfy the constraints. Therefore, the discretized solution is always feasible, but the continuous solution might not be feasible [36]. One solution to this problem is to use more nodes. The optimal path will then be less likely to be infeasible or cut the corner of an obstacle, but this causes increased computational complexity. Another solution is, as illustrated in Figure 6.10 to fictitiously enlarge the size of the obstacle when describing the path constraint [36].

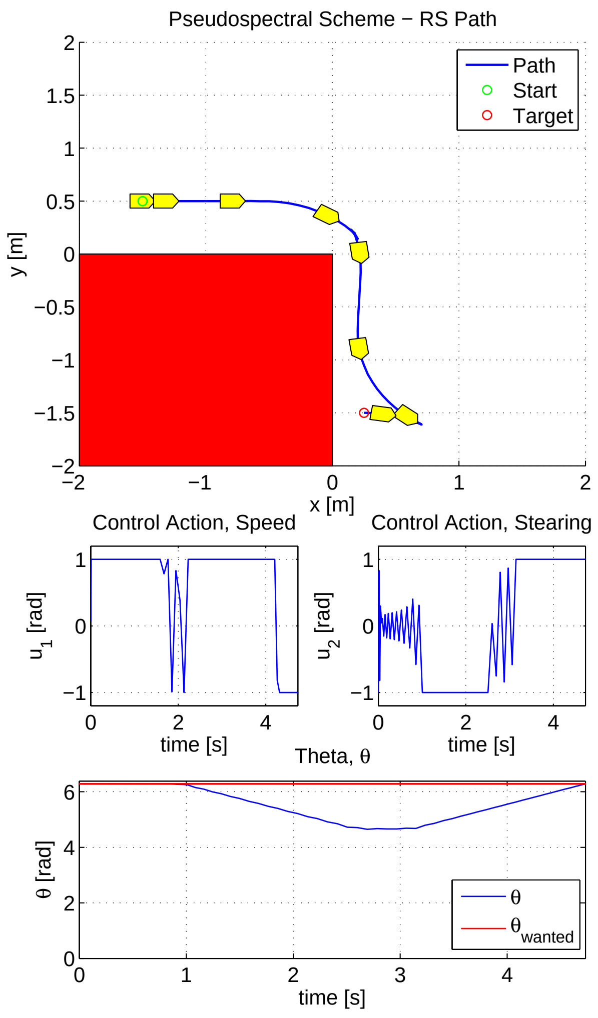
The tests run for a Reeds-Shepp car with obstacles are shown in figures 6.10 and 6.11. The corresponding test values and run conditions are shown in Table 6.4.

Initial pose	Final pose	N	Domain Ω	ρ_{min}	Obstacle Clearance	Figure
(-1.5, 0.5, 2π)	(0.25, -1.5, 2π)	80	$[-2,2] \times [0,2\pi)$	1	0	6.10a
(-1.5, 0.5, 2π)	(0.25, -1.5, 2π)	80	$[-2,2] \times [0,2\pi)$	1	0.2	6.10b and 6.11a
(-1.5, 0.5, 2π)	(0.25, -1.5, 2π)	30	$[-2,2] \times [0,2\pi)$	1	0.2	6.11b

Table 6.4: Test Values for Optimized Paths for RS with Obstacles Using PS Method

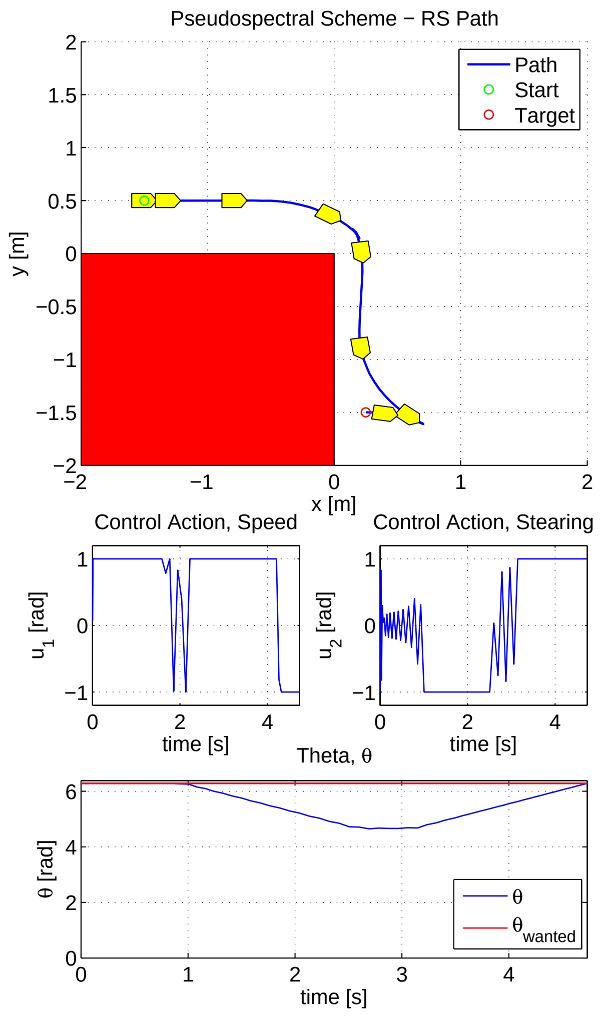


(a) Optimized Paths Regular Obstacle Area

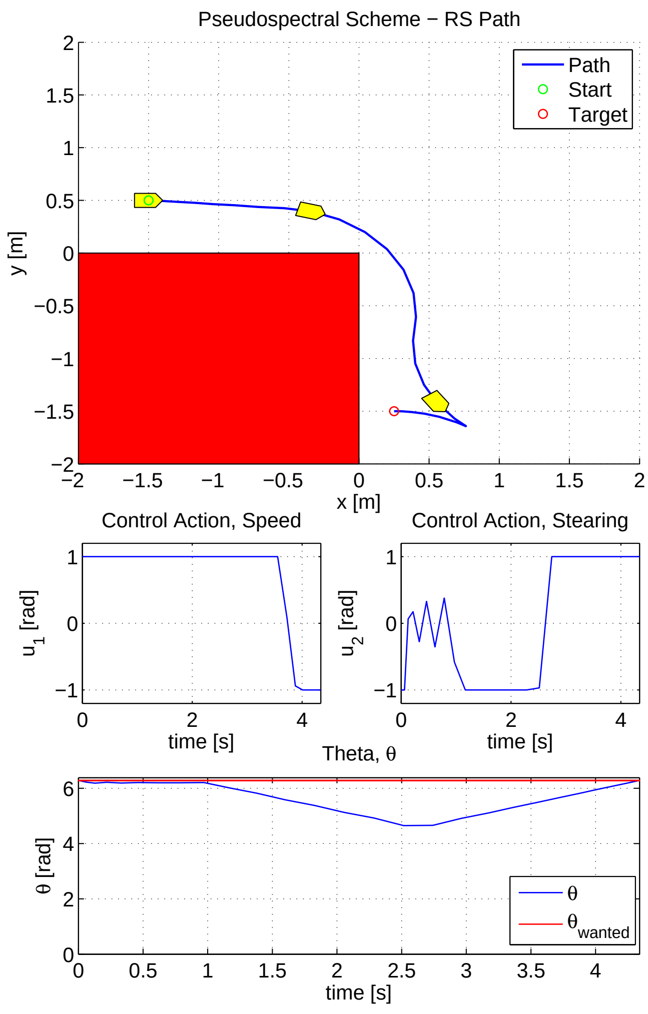


(b) Optimized Paths Expanded Obstacle Area

Figure 6.10: RS Car with Obstacles using PS - Comparing Obstacle Clearance. Optimized path from $(-1.5, 0.5, 2\pi)$ to $(0.25, -1.5, 2\pi)$



(a) Optimized Paths with N = 80



(b) Optimized Paths with N = 30

Figure 6.11: RS Car with Obstacles using PS - Comparing N Values. Optimized path from $(-1.5, 0.5, 2\pi)$ to $(0.25, -1.5, 2\pi)$

6.3 Boat Model

This section describes the calculations and results when solving the optimal control problem by using the MATLAB application DIDO and the PS method to produce the shortest path for a simplified boat model. The theory used to conduct the implementation is found in sections 3.4 and 4.2.

The model used in this section includes both surge and sway velocities and is described by (4.9)-(4.13) in Section 4.2. By redefining the boundaries on the state and controls of the system and setting up the code structure again, this time for the simplified boat model, DIDO was ready to calculate the optimized path.

A selection of the shortest length paths given by this implementation is shown in figures 6.12 to 6.14. The corresponding test values and run conditions are shown in Table 6.5. It is seen that the boat takes advantage of the different directional speeds to produce the shortest path possible, thus giving it an even shorter path then the Dubins car. This can be seen by comparing figures 6.12 and 6.1.

Initial pose	Final pose	N	Domain Ω	ρ_{min}	Figure
(1,1,$-\frac{\pi}{2}$)	(-1,-1,$\frac{\pi}{2}$)	60	$[-2,2] \times [0,2\pi)$	0.5	Figure 6.12
(0.5, 1, $-\frac{\pi}{2}$)	(-0.5, 1, $\frac{\pi}{2}$)	60	$[-2,2] \times [0,2\pi)$	0.5	Figure 6.13
(0, 1.5, 0)	$(0, -1.5, \pi)$	60	$[-2,2] \times [0,2\pi)$	0.5	Figure 6.14

Table 6.5: Test Values for Optimized Paths for Boat Model Using PS Method

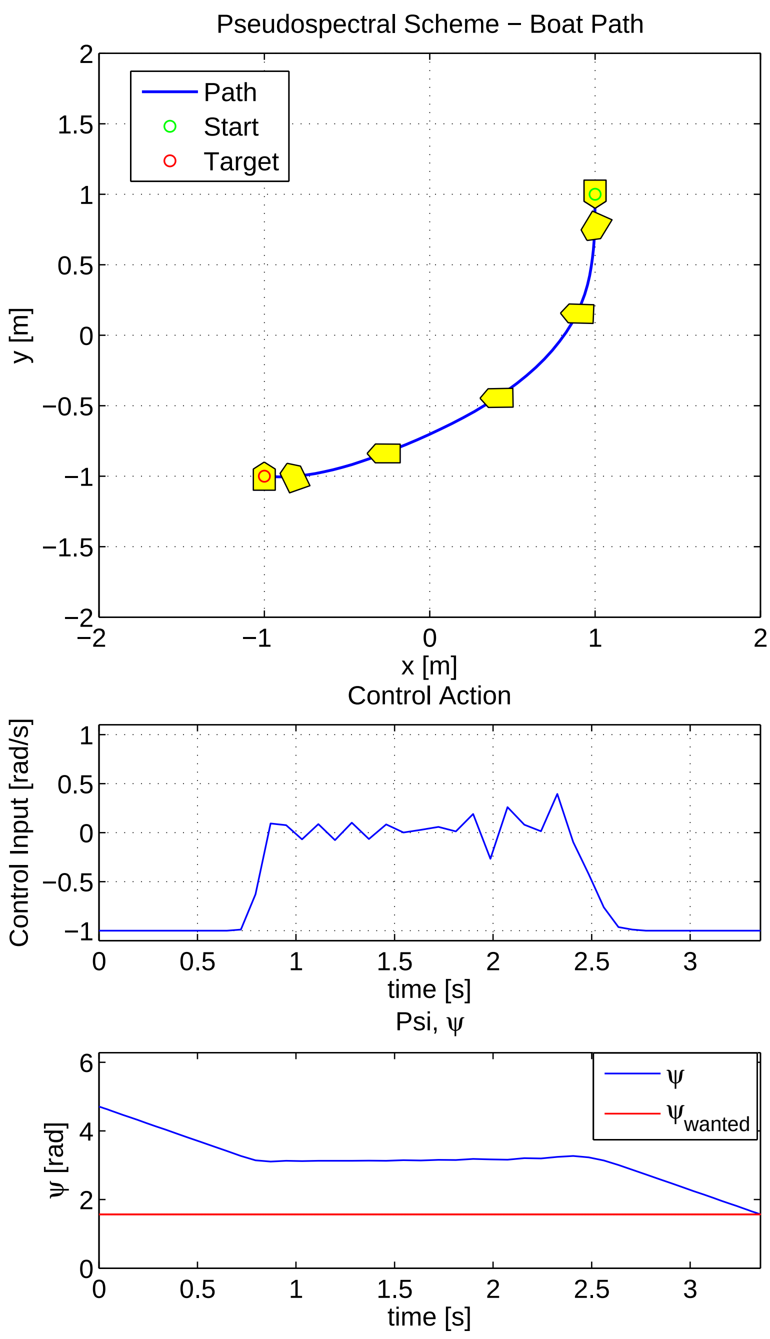


Figure 6.12: Boat Using PS - Optimized Path from $(1, 1, -\frac{\pi}{2})$ to $(-1, -1, \frac{\pi}{2})$

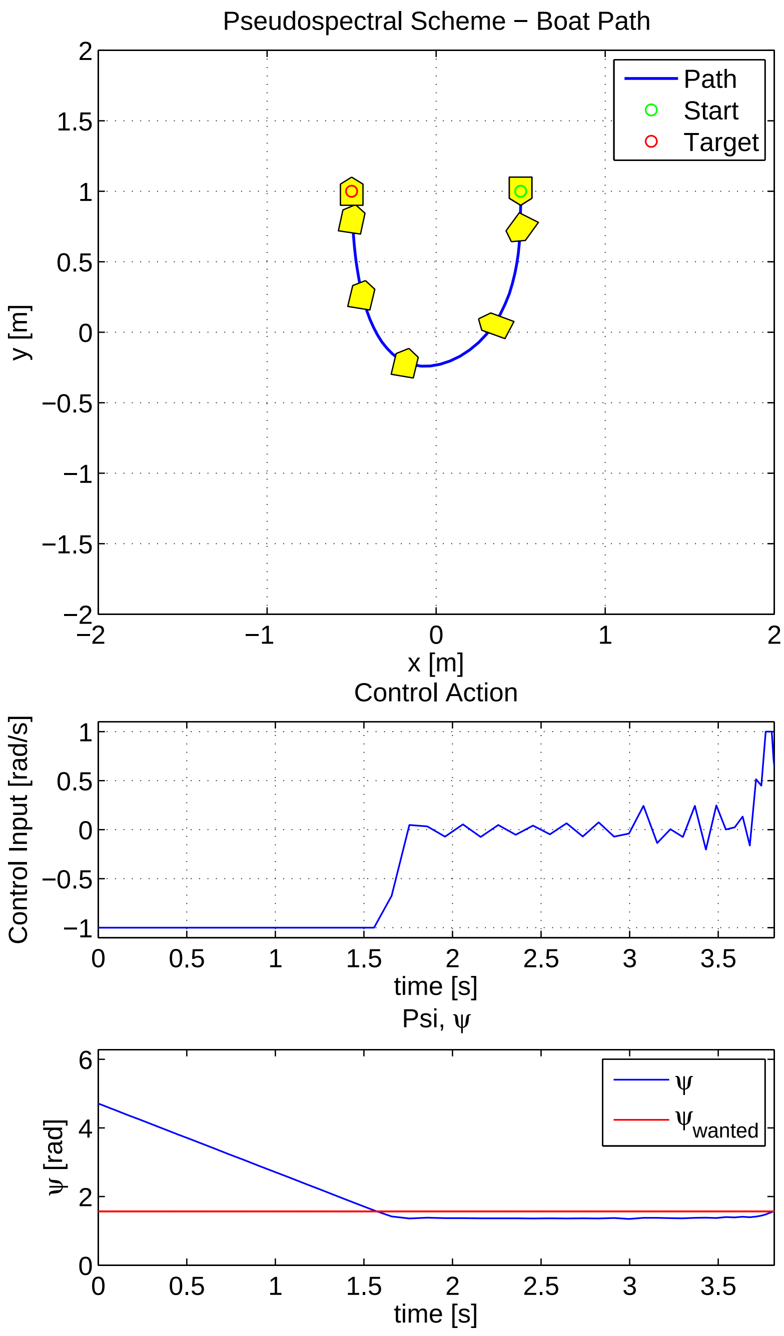


Figure 6.13: Boat Using PS - Optimized Path from $(0.5, 1, -\frac{\pi}{2})$ to $(-0.5, 1, \frac{\pi}{2})$

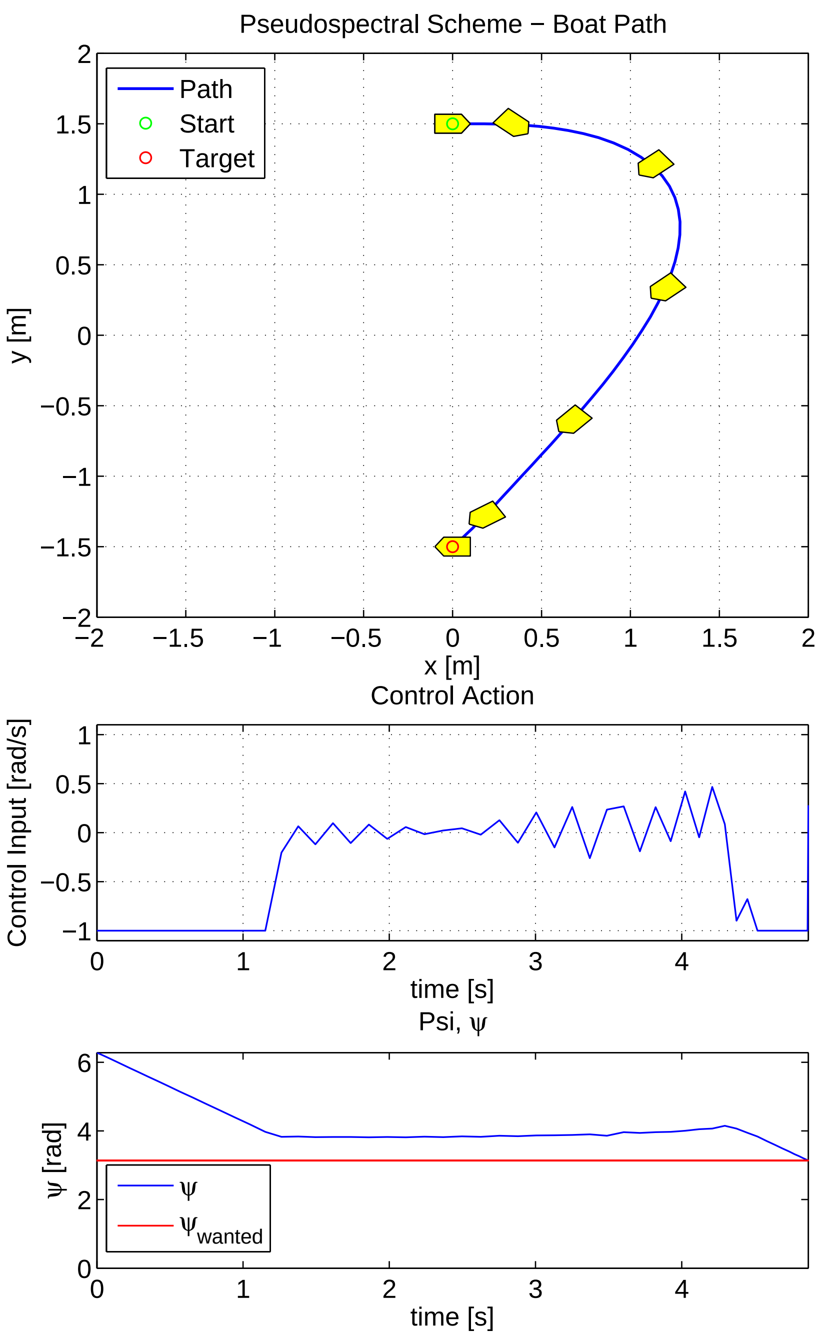


Figure 6.14: Boat Using PS - Optimized Path from $(0, 1.5, 0)$ to $(0, -1.5, \pi)$

Chapter 7

Boat Model with Ocean Current

In the previous two chapters, the numerical optimization methods SL and PS were tested on different models, in different environments and with different start and end poses. When studying the results, it is clear that the PS method is both faster and gives more accurate results. Therefore, this method is used in the remaining implementations and case studies in this thesis.

This chapter describes the calculations and results given when solving the optimization problem by using the MATLAB application DIDO and the PS method to produce the shortest path for a simplified boat model affected by ocean current. The theory used to conduct the implementation, being PS method and the boat model, is found in sections 3.4 and 4.2.

Boats are affected by the ocean current that surrounds them, but often the ocean current is unknown. Adding this factor to the boat model used in Section 6.3 and introducing the terms relative surge velocity, $u_r = u - u_c$, and relative sway velocity, $v_r = v - v_c$, results in the model used in this chapter and described by (4.25) - (4.29).

Relevant guidance theory and the specific equations used in the implementation for this thesis are presented first in this chapter. Thereafter, it is shown that the optimal path found by the PS method does not compensate or take into consideration the unknown current affecting the boat. Therefore, in the first tests the boat is unable to follow the optimal path given by the PS method, and is instead drawn away from the desired path by the current. A guidance system and adaptive observers are then used to try to compensate for this error. This is done using two different approaches: initial path planning with continuous current estimation and repeated path planning. The PS method is used first to calculate one initial optimal path, assuming no ocean current. The initial path is used as a reference path. This reference path is used as the input to the guidance system, which will try to estimate the ocean current and use this information to make the boat converge towards the desired path. The second approach is to use the guidance system to estimate the ocean current and then use this estimate together with PS to continuously calculate new optimal paths based on new ocean current information. With time, the PS method will have more correct information about the current, and will therefore be able to calculate a path that compensates for this current. This causes the actual boat to be able to follow the updated optimized path with no speed or heading corrections.

7.1 Adaptive Guidance Techniques

Guidance systems are very important when considering the performance of a vehicle and how well the vehicle is able to follow or track the reference path or positions given by the path planner. General background information about the guidance system and its role in the motion control system is presented in Section 2.1. The most important capability of the guidance system is, however, that it provides the vehicle's controller with reference trajectories for either keeping the vehicle on the path or, if the position error is nonzero, leading it towards the path [32]. Some of the most popular methods adopted by the marine community are LOS guidance (see also [18] and [6]), pure pursuit guidance and constant bearing guidance. All of these methods are presented in [5]. In [18], an LOS guidance law for straight line path following is implemented, and in [6] the same method is implemented for curved path following.

7.1.1 Path Following and Trajectory Tracking

Within guidance, there are various motion control scenarios and goals. Two motion control scenarios are path following and trajectory tracking. The main difference between path following and trajectory tracking is that path following is time-invariant, while trajectory tracking is time-varying [17]. More specifically, path following refers to the case where a vehicle follows a predefined path independent of time. As a result, this type of reference trajectory has no velocity constraints and the guidance system therefore only generates reference commands for the heading. For trajectory tracking, on the other hand, the objective is for the vehicle to follow a desired path $\mathbf{x}_d(t)$ with both spatial and temporal constraints [35]. Consequently, the vehicle has to have a specific position at each time instance. For more details on various motion control scenarios, see [5] and [7].

7.1.2 Trajectory Tracking and Ocean Current Estimation

In sections 7.2 and 7.3 of this thesis, a guidance system for trajectory tracking and ocean current estimation is implemented, as described in detail in [35]. It is run in combination with the optimization technique PS, in order to produce short, feasible paths as well as to estimate and counteract the unknown ocean current.

The total position error vector has two components. These are the along-track error, which is tangent to the path, and the cross-track error, which is normal to the path. For path following, the important component to focus on is the cross-track error, while for trajectory tracking, both components are important for convergence. In [35] it is shown how adaptive nonlinear observers are used to estimate the current components, $\hat{\theta}_y$ and $\hat{\theta}_x$. The current components, $\hat{\theta}_y$ and $\hat{\theta}_x$, are defined as the normal and tangent ocean current components w.r.t. the body-fixed reference frame and are therefore constant for a boat with constant heading. This information is used by a guidance algorithm (described in detail in [34]) to generate relative surge speed reference trajectories for minimizing the along-track error. Simultaneously, a version of the Line-Of-Sight (LOS) guidance (complete proofs in [19]) is used to minimize the cross-track error. In this section the main equations and techniques from [35] used for this implementation are repeated. The main elements and variables used in LOS guidance are illustrated in Figure 7.1.

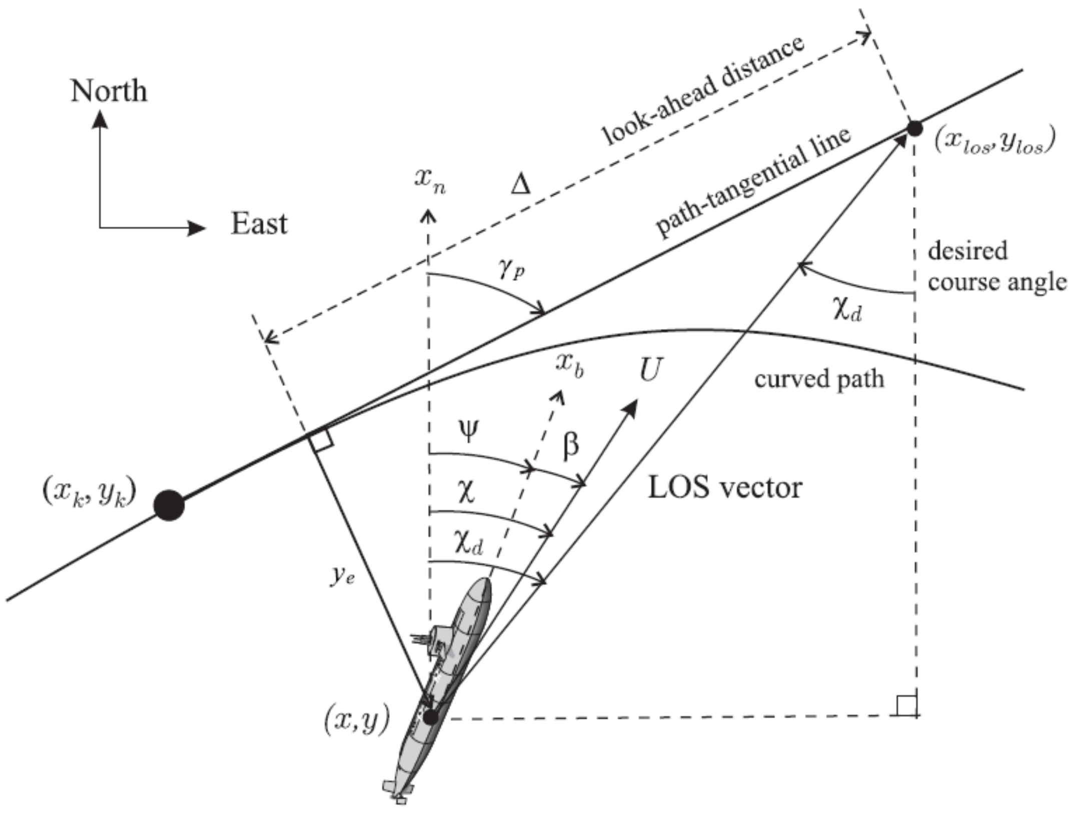


Figure 7.1: Line-Of-Sight Guidance for Curved Paths [33]

The cross-track error y_e and along-track error x_e are given by equations (7.1) and (7.2). This is also illustrated in Figure 7.2 in relation to two different desired positions.

$$y_e = -(x - x_d)\sin(\gamma_p) + (y - y_d)\cos(\gamma_p) \tag{7.1}$$
$$x_e = (x - x_d)\cos(\gamma_p) + (y - y_d)\sin(\gamma_p) \tag{7.2}$$

In (7.1) and (7.2), x_d and y_d represent the desired position at the current time, and γ_p represents the path-tangential angle (also illustrated in Figure 7.1).

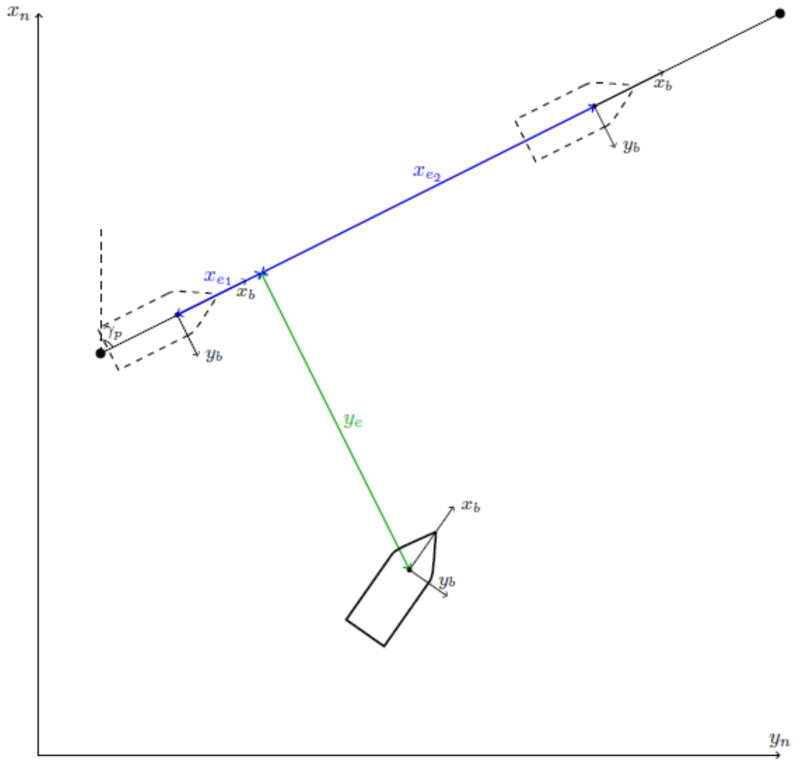


Figure 7.2: Cross-Track and Along-Track Error [34]

The adaptive observers are $\hat{y}_e, \hat{x}_e, \hat{\theta}_y$ and $\hat{\theta}_x$ and are updated by [35]

$$\dot{\hat{y}}_e = -\frac{U_r(\hat{y}_e + \alpha_y)}{\sqrt{\Delta^2 + (y_e + \alpha_y)^2}} + \hat{\theta}_y + k_1(y_e - \hat{y}_e) \tag{7.3}$$

$$\dot{\hat{\theta}}_y = k_2(y_e - \hat{y}_e) \tag{7.4}$$

$$\dot{\hat{x}}_e = -k_x\hat{x}_e + \hat{\theta}_x + \alpha_x + k_3(x_e - \hat{x}_e) \tag{7.5}$$

$$\dot{\hat{\theta}}_x = k_4(x_e - \hat{x}_e) \tag{7.6}$$

where k_i represents tuning constants, U_r is the relative speed and Δ is the user specified lookahead distance of the LOS guidance scheme. The lookahead distance is also illustrated in Figure 7.1. α_x and α_y are control inputs designed to be:

$$\alpha_x = -\hat{\theta}_x \tag{7.7}$$

$$\alpha_y = \Delta\frac{(\hat{\theta}_y/(U_r)))}{\sqrt{1 - (\hat{\theta}_y/U_r)^2}} \tag{7.8}$$

The current components, θ_y and θ_x which are estimated by the observers, can also be used to compute the ocean current vector [35]. This is done by the following equations:

$$\theta_y = U_c \sin(\beta_c - \gamma_p) \tag{7.9}$$

$$\theta_x = U_c \cos(\beta_c - \gamma_p) \tag{7.10}$$

$$\Downarrow \tag{7.11}$$

$$\hat{U}_c = \sqrt{\hat{\theta}_y^{\,2} + \hat{\theta}_x^{\,2}} \tag{7.12}$$

$$\hat{\beta}_c = \gamma_p + \arctan(\frac{\hat{\theta}_y}{\hat{\theta}_x}) \tag{7.13}$$

where $\hat{(\cdot)}$ represents the estimates given by the observers. In Section 7.2.2, this estimate is used to produce new optimized paths based on more accurate information about the ocean current.

As mentioned above, the estimates from the adaptive observers are used by the guidance system to generate wanted relative speed, u_d, and heading, ψ_d, for minimizing the along-track and cross-track error. With no environmental disturbances the LOS guidance is represented by $\psi_d = \gamma_p - \beta_r + \arctan(-\frac{y_e}{\Delta})$. Since unknown external forces are present, the LOS is augmented to include a control action α_y which can be designed to produce integral action. The equations used are the following [35]:

$$\psi_d = \gamma_p - \beta_r + \arctan\left(-\frac{1}{\Delta}(y_e + \alpha_y)\right) \tag{7.14}$$

$$u_d = \sqrt{1+\xi^2}\left(-v_r \frac{\xi}{\sqrt{1+\xi^2}} - k_x x_e + U_t + \alpha_x \right) \tag{7.15}$$

where

$$\xi = \frac{v_r \Delta + u_r(y_e + \alpha_y)}{v_r(y_e + \alpha_y) - u_r \Delta} \tag{7.16}$$

$$\beta_r = \arctan(\frac{v_r}{u_r}) \tag{7.17}$$

Here u_r and v_r represent the relative speed in surge and sway direction and U_t is the target speed, which is the "object" being tracked.

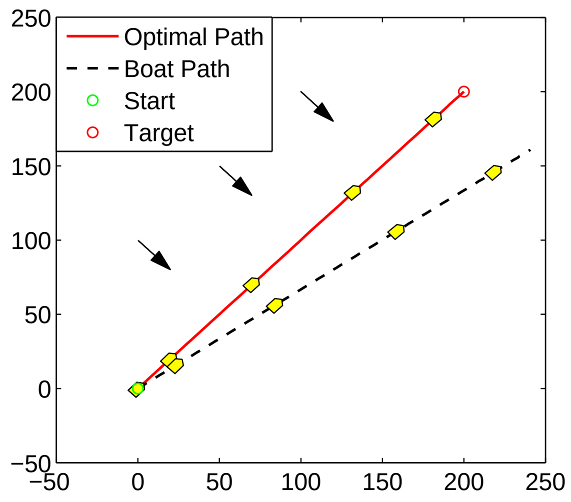
7.2 Boat Model with Ocean Current

In this section two different techniques for compensating for unknown ocean current are presented and implemented. All of the testes in this chapter are run with 60 nodes and within the domain $\Omega = [0, 200] \times [0, 2\pi)$. The initial and final poses and the remaining run conditions for all tests are shown in Table 7.1.

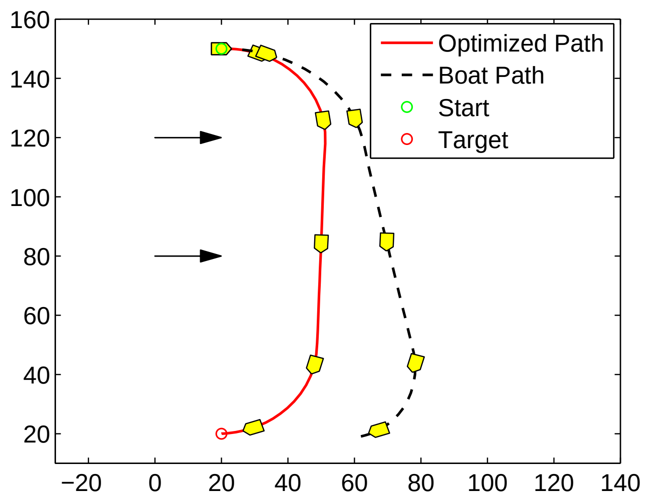
Test case	Initial pose	Final pose	ρ_{min}	Max speed	V_x, V_y
1	$(0,0,\frac{\pi}{4})$	$(200,200,\frac{\pi}{4})$	1	5	0.7071 m/s, -0.7071 m/s
2	(20,150,0)	$(20,20,\pi)$	30	0.5	0.1 m/s, 0 m/s

Table 7.1: Test Values for Optimized Paths for a Boat with Ocean Current

When using PS to calculate an optimized path when there is an unknown disturbance present, the path calculated does not compensate for this disturbance. By using the optimal control estimated by PS directly, without the use of any additional guidance, observers or control systems, the boat is unable to follow the desired path or reach its target. This is seen in figures 7.3a and 7.3b.



(a) Optimized Path from $(0, 0, \frac{\pi}{4})$ to $(200, 200, \frac{\pi}{4})$



(b) Optimized Path from $(20, 150, 0)$ to $(20, 20, \pi)$

Figure 7.3: Boat Paths with Unknown Ocean Current

7.2.1 Initial Path Planning and Continious Current Estimation

As seen in the previous section, the boat is not able to follow the desired path or reach its target as a result of the unknown current. The goal is therefore to try to remove this error by implementing adaptive guidance and observers that estimate the ocean current, and to then use this information to reduce the errors that prevent the boat from converging towards the optimal path. The theory for the guidance algorithms and observers implemented is presented in Section 7.1.

As shown in Table 7.1, for test case 1 the virtual vehicle being tracked moves with speed $U_t = 5$ m/s along the line connecting the waypoints (0, 0) and (200, 200). This gives the path-tangential angle $\gamma_p = 45°$ C. The ocean current velocity vector magnitude is $U_c = 1$ m/s and its orientation w.r.t. the inertial frame is $\beta_c = -45°$ C. In addition, the test is run with lookahead distance $\Delta = 60$ m and gain values $k_x = 0.05$, $k_1 = 6$, $k_2 = 0.8$, $k_3 = 5$ and $k_4 = 1$.

For test case 2 the virtual vehicle moves with speed $U_t = 0.5$ m/s along a curved path. The ocean current velocity vector magnitude is $U_c = 0.1$ m/s and its orientation w.r.t. the inertial frame is $\beta_c = 0°$ C. The lookahead distance for the LOS algorithm is $\Delta = 10$ m and the gain values are $k_x = 0.05$,$k_1 = 10$, $k_2 = 2.5$, $k_3 = 10$ and $k_4 = 5$. For both tests no prior estimation or measurement of the ocean current is available, therefore the observers' initial conditions are $(\hat{x}_e, \hat{\theta}_x, \hat{y}_e, \hat{\theta}_y)^T = (0, 0, 0, 0)^T$.

In figures 7.4 and 7.5 the results from test case 1 and 2 are illustrated. The red line represents the optimized path calculated using PS and assuming no ocean current. The blue line is the actual boat with guidance system and observers implemented to counteract the unknown current, while the black line is the actual boat path with no guidance system. It can be seen that the boat with guidance converges towards the path and thereafter keeps tracking the optimized path until it reaches the target point. Although the guidance system is able to adjust the heading of the boat to counteract the unknown current and keep the desired position, the boat is not always able to reach the target with desired target orientation.

In figures 7.6 and 7.7 the guidance system results are shown by plotting various error estimates and current estimates together with real values.

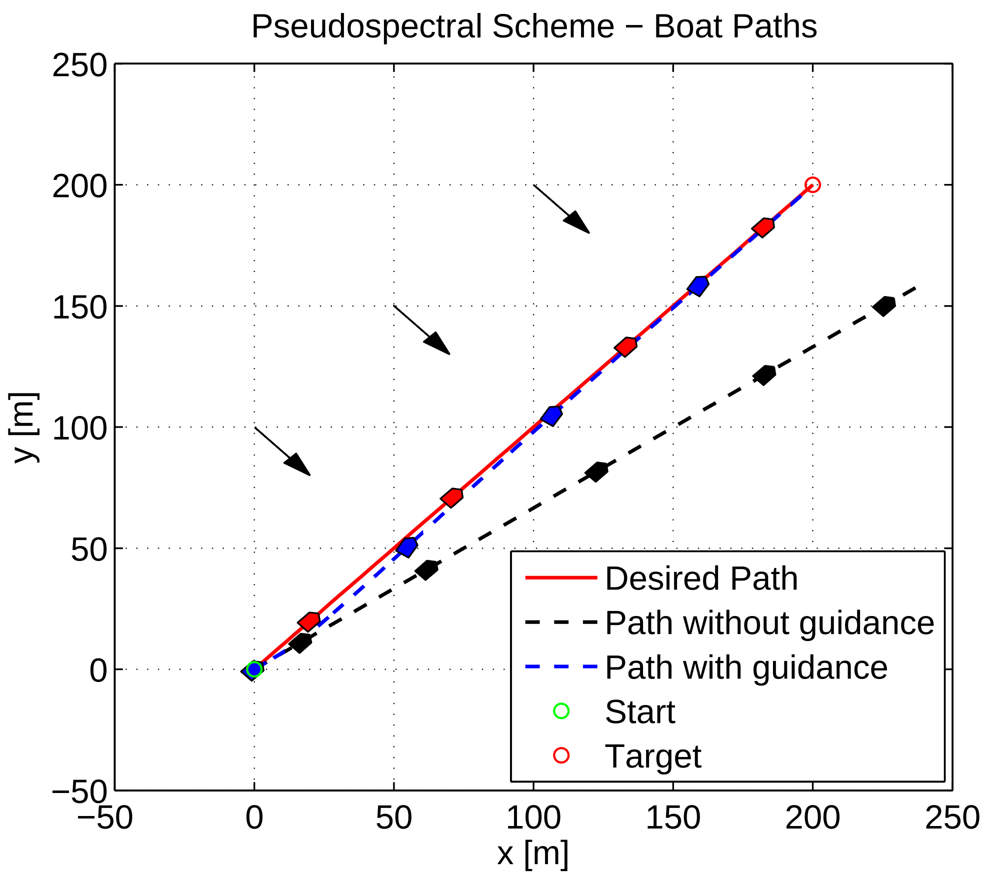


Figure 7.4: Straight Boat Path with Ocean Current Using PS and Adaptive Guidance. Optimized path from $(0, 0, \frac{\pi}{4})$ to $(200, 200, \frac{\pi}{4})$

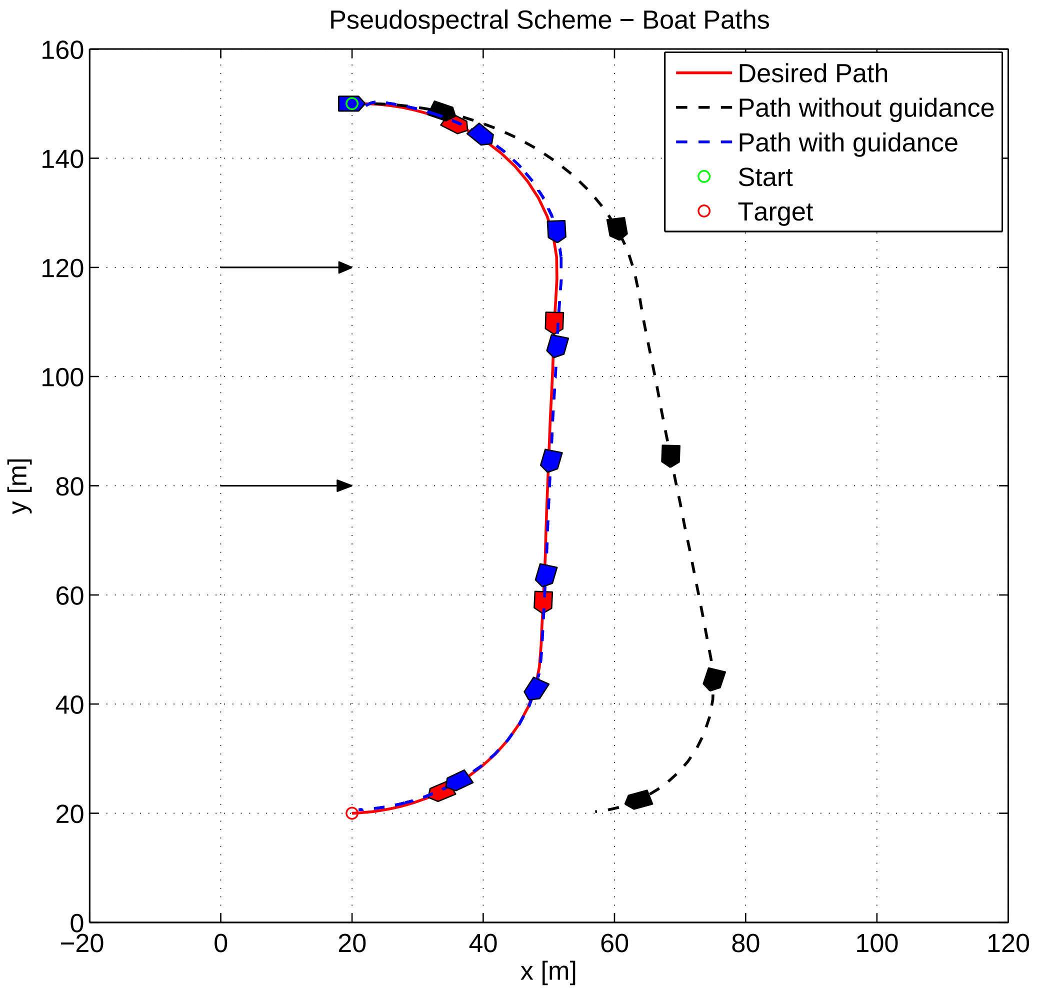
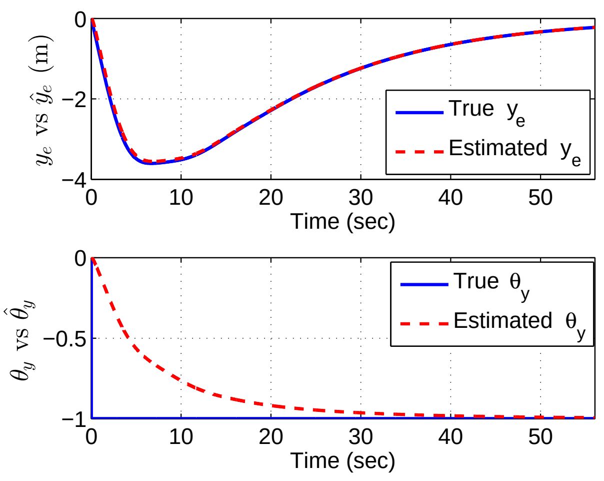
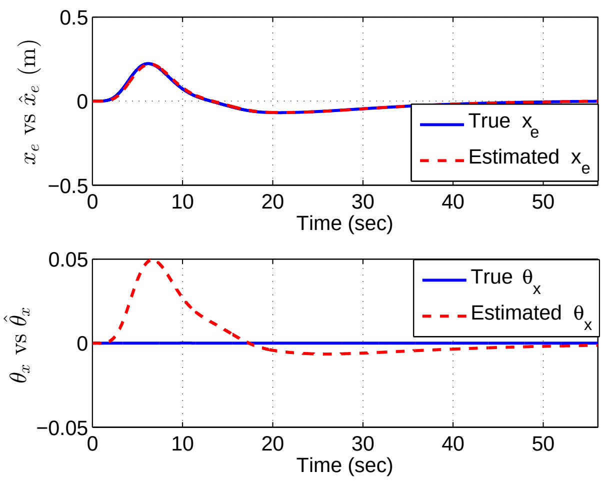


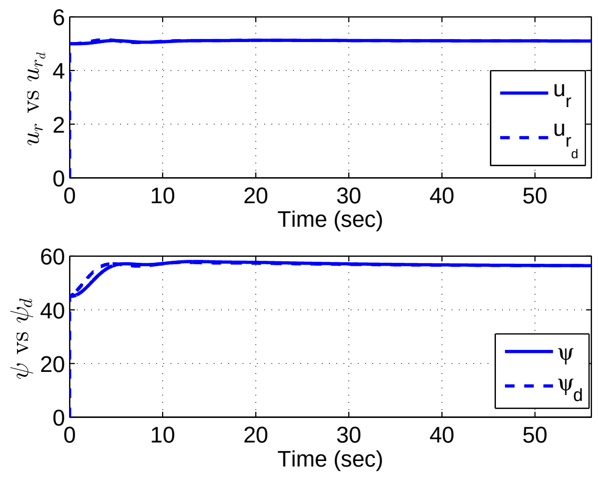
Figure 7.5: Curved Boat Path with Ocean Current Using PS and Adaptive Guidance. Optimized path from $(20, 150, 0)$ to $(20, 20, \pi)$



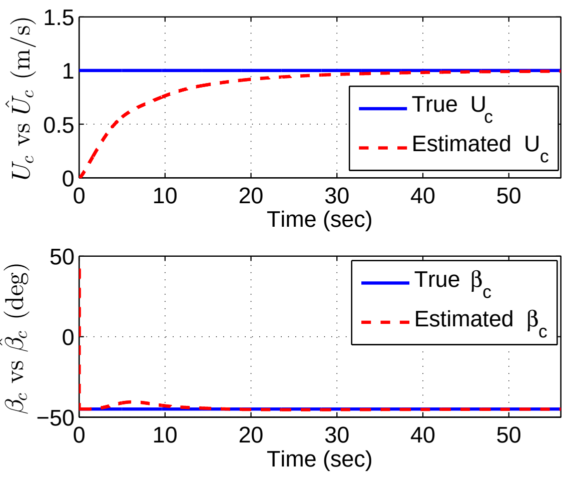
(a) y_e vs. $\hat{y}_e$ and θ_y vs. $\hat{\theta}_y$



(b) x_e vs. $\hat{x}_e$ and θ_x vs. $\hat{\theta}_x$

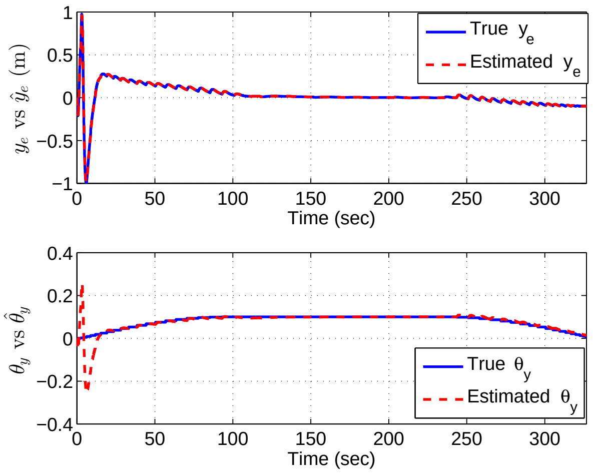


(c) u_r vs. u_{r_d} and ψ vs. ψ_d

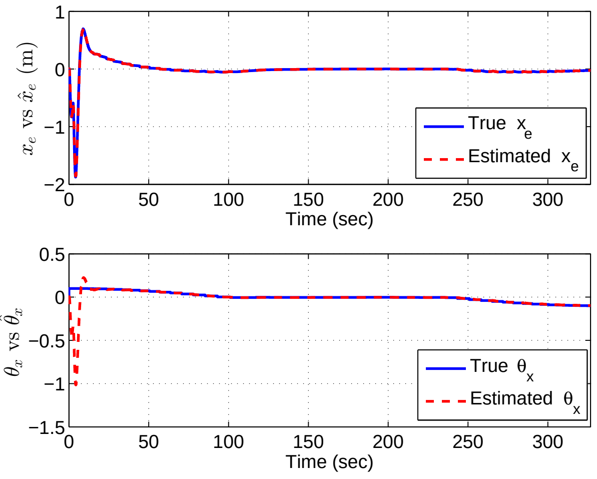


(d) U_c vs. $\hat{U}_c$ and β_c vs. $\hat{\beta}_c$

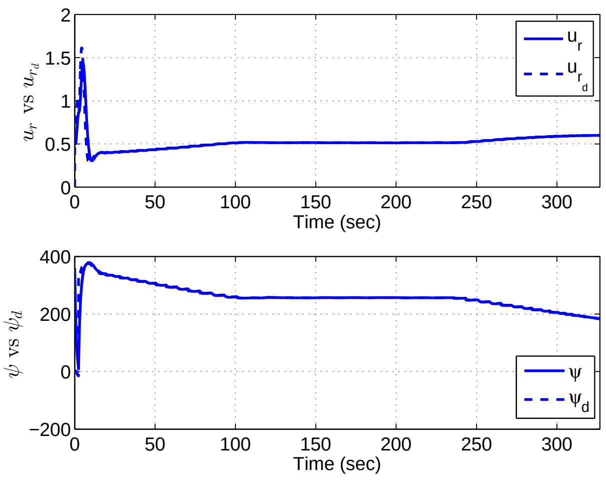
Figure 7.6: Guidance Results for Initial Path Planning Test Case 1



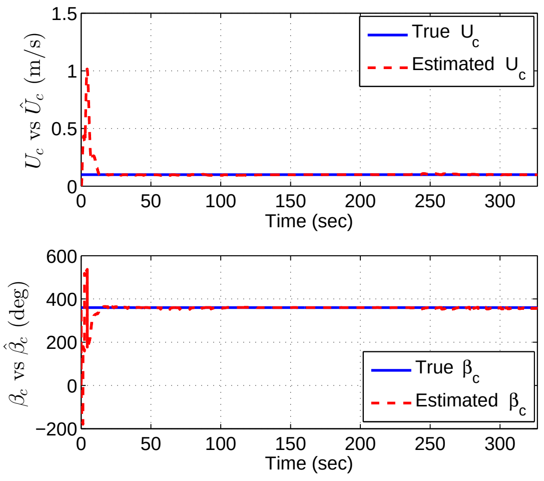
(a) y_e vs. $\hat{y}_e$ and θ_y vs. $\hat{\theta}_y$



(b) x_e vs. $\hat{x}_e$ and θ_x vs. $\hat{\theta}_x$



(c) u_r vs. u_{r_d} and ψ vs. ψ_d



(d) U_c vs. $\hat{U}_c$ and β_c vs. $\hat{\beta}_c$

Figure 7.7: Guidance Results for Initial Path Planning Test Case 2

7.2.2 Repeated Path Planning Using Current Estimation

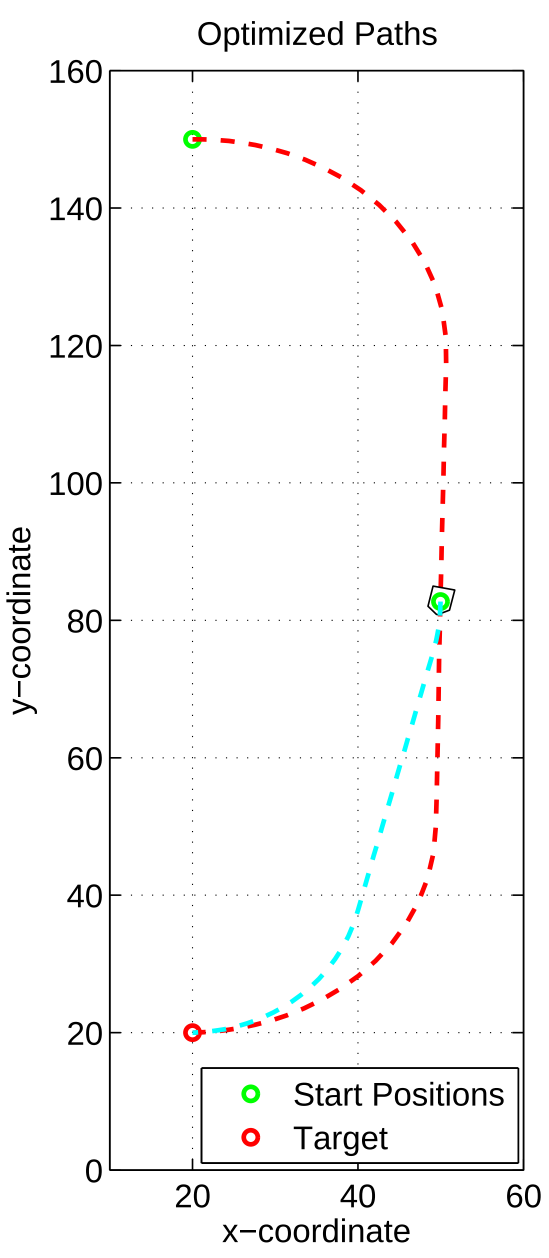
The second approach tested is the use of the PS method and the guidance system simultaneously, therefore performing a type of repeated path planning. The PS method runs with even intervals and for each run it uses the new estimates of V_x and V_y given by the guidance system. By implementing the system in this manner, the optimized path changes while the boat moves, and the newest and most correct estimate of the current is used to calculate new and more feasible paths. The theory for the guidance algorithms and observers implemented are presented in Section 7.1. This approach was implemented with a convergence test on the current estimation. After sufficient convergence is detected for both U_c and β_c, both values are constant for the rest of the run. For both tests illustrated below, the current estimates converge before the path is updated the first time.

In Figure 7.9, the optimized paths for two different runs are shown. In 7.9a, a new optimized path is calculated every 170 seconds, and in 7.9b a new path is calculated every 90 seconds. The start position for each new optimized path is marked with a green dot. For this implementation, the starting point for each new optimized path is set to be the current position of the real boat, not the current desired position given by the optimized path. As a result, the final optimized path is smoother, with fewer sudden jumps and quick position changes. In Figure 7.8, the optimized paths when using these two different starting poses are shown. It is seen that when using the current desired pose as a starting pose, the path is more uneven. The reason for this is that the heading and speed of the desired pose does not necessarily compensate for any unknown disturbances. This causes the new optimized path, which has more knowledge of the current, to first spend time changing its heading before it is able to move in the right direction again.

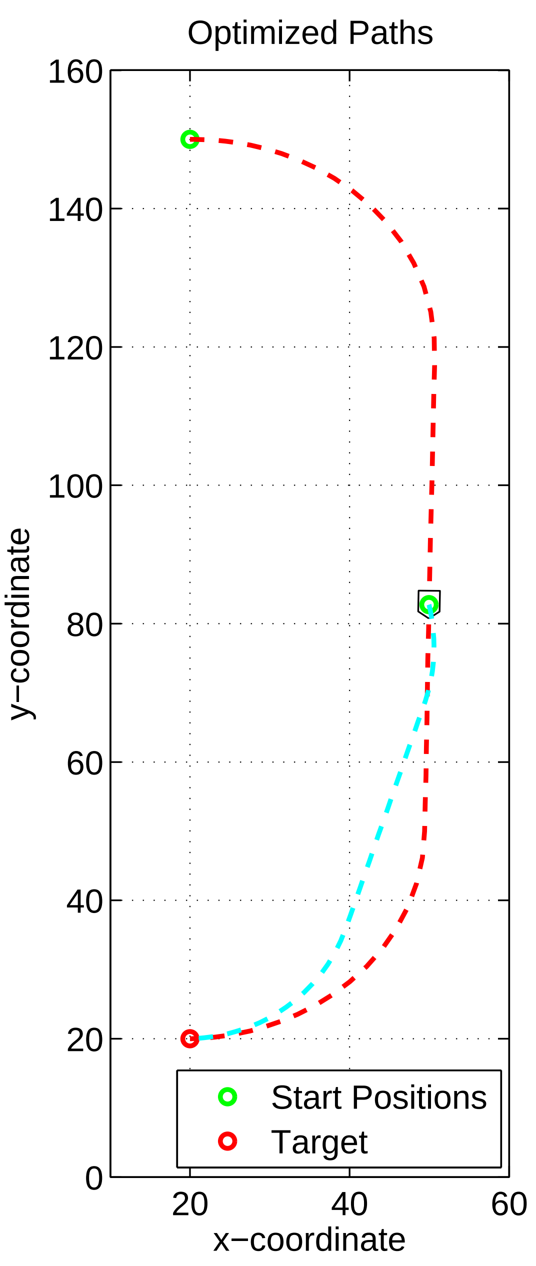
In Figure 7.9, the individual optimized paths are plotted with different colors to make it easier to distinguish between them. In Figure 7.10, the corresponding boat paths are plotted with the final optimized path. It is seen that the guidance system, with the update of the desired optimized path, is able to adjust the heading of the boat to counteract the unknown current and keep the desired position along the entire path. In addition, the heading of the real boat and the optimized path are similar after the optimized path is updated. This is due to the fact that the desired path counteracts the estimated current by changing its heading. The optimized path is therefore a more feasible path and the real boat does not have to adjust the speed and orientation to track the path. This also causes the boat to reach its target with desired orientation, which was not necessarily the case with the initial path planning approach presented in Section 7.2.1.

For both tests, the virtual vehicle being tracked moves with speed $U_t = 0.5$ m/s along the curved path. The ocean current velocity vector magnitude is $U_c = 0.1$ m/s and its orientation w.r.t. the inertial frame is $\beta_c = 0°$ C. The lookahead distance for the LOS algorithm is $\Delta = 10$ m. The gain values are $k_x = 0.2$, $k_1 = 10$, $k_2 = 2.5$, $k_3 = 10$ and $k_4 = 5$. There is no prior estimation or measurement of the ocean current available, therefore the observers' initial conditions are $(\hat{x}_e, \hat{\theta}_x, \hat{y}_e, \hat{\theta}_y)^T = (0, 0, 0, 0)^T$.

In figures 7.11 and 7.12, the guidance system results are shown by plotting different error estimates and current estimates together with the real values. It is seen that all estimates converge towards the true value, although the along-track error does not converge to zero. This is a numerical issue that must be further investigated. For the test with $t_{\text{new path}} = 90$, the current estimate deviates slightly from the true value, causing the cross-track error y_e to also have a slight deviation from zero. For $t_{\text{new path}} = 170$, the convergence criteria is set to a lower value in order to get a more accurate current estimate, and thereby removing the deviations in y_e. This is possible because the time before the first update is higher, giving the estimates more time to converge properly.

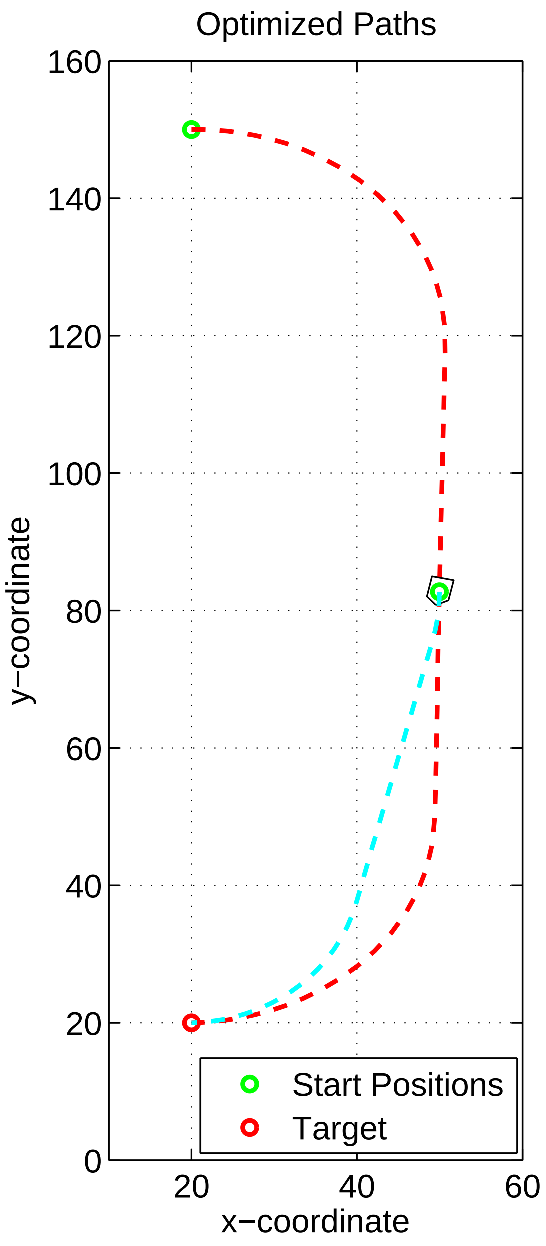


(a) Start = Real Boat Position

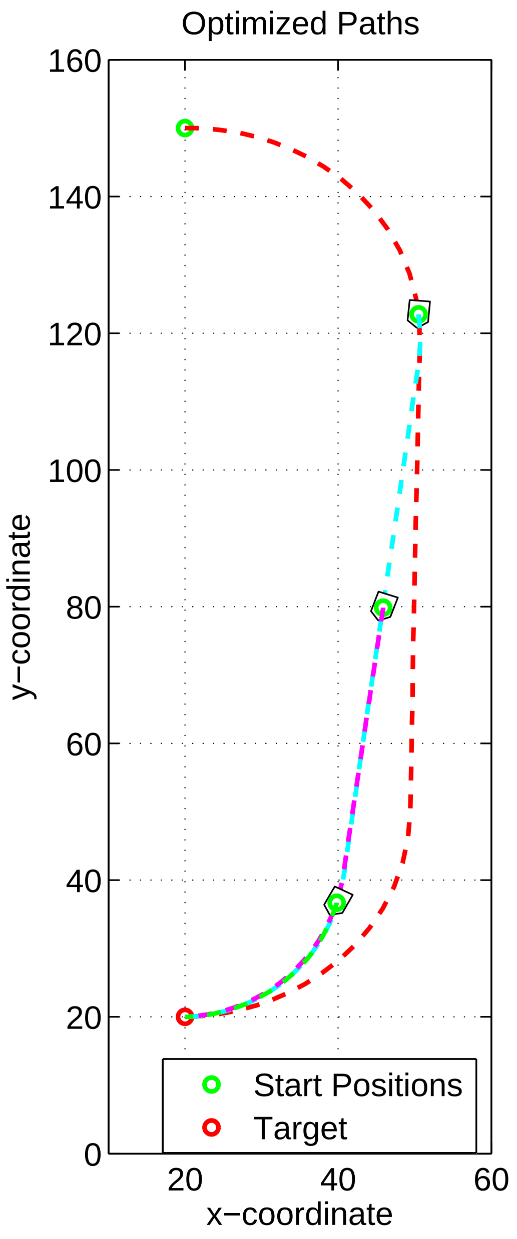


(b) Start = Wanted Position

Figure 7.8: Optimized Paths with Different Start Positions

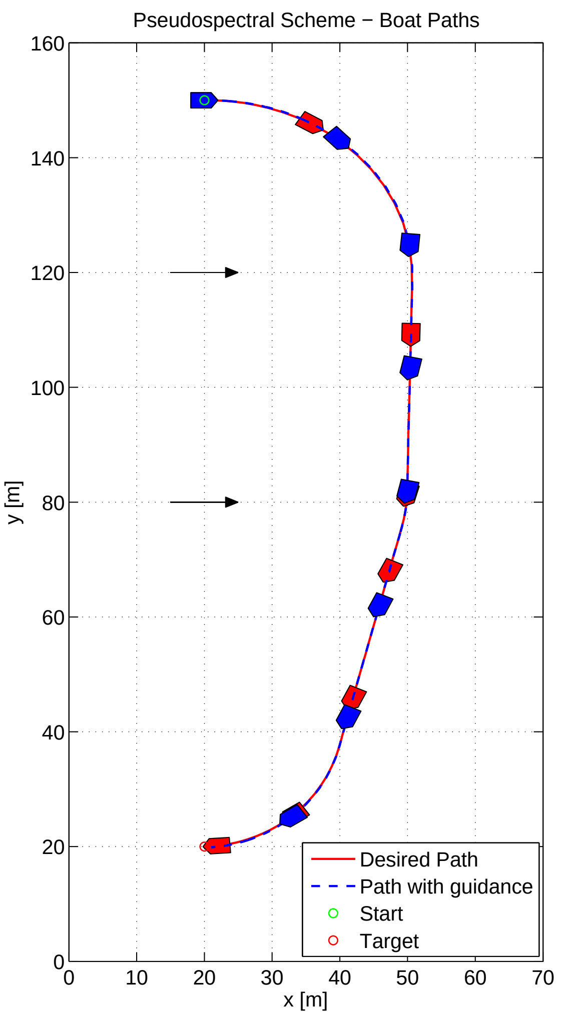


(a) Optimized Paths for $t_{\text{new path}} = 170$

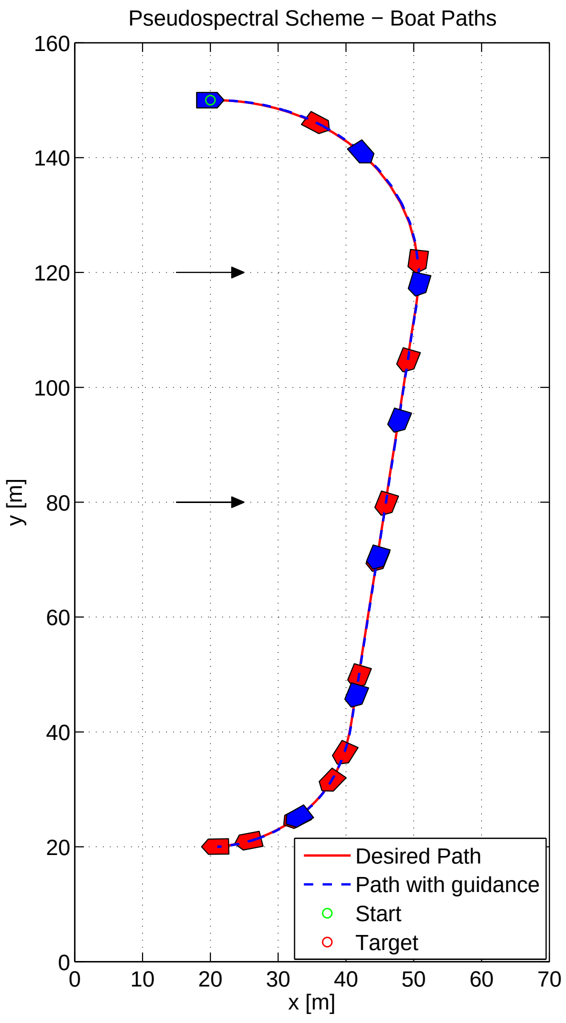


(b) Optimized Paths for $t_{\text{new path}} = 90$

Figure 7.9: Optimized Paths Updated at Different Time Intervals

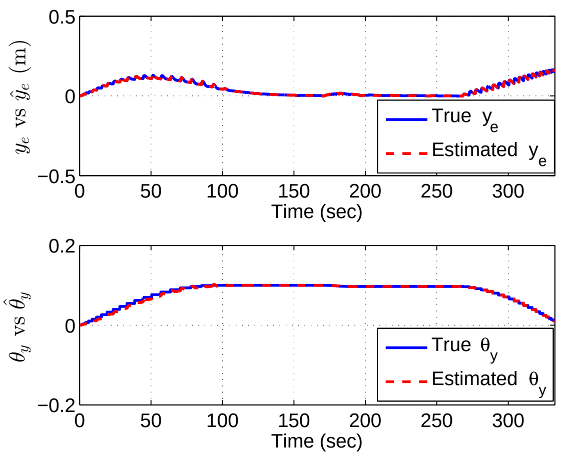


(a) Optimized Paths for $t_{\text{new path}} = 170$

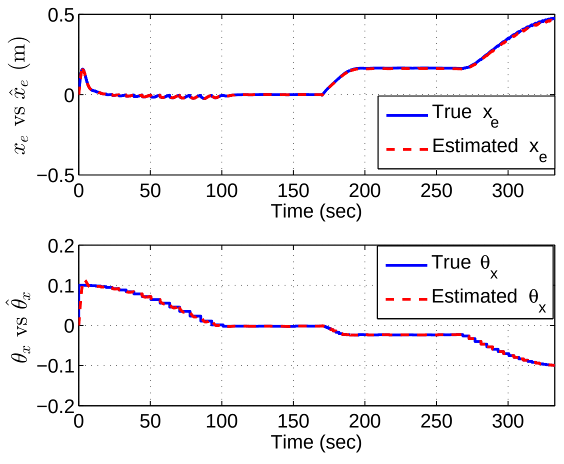


(b) Optimized Paths for $t_{\text{new path}} = 90$

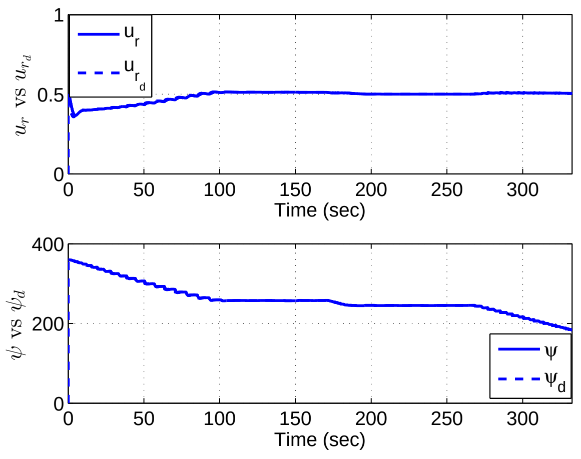
Figure 7.10: Boat Paths with Ocean Current Using Repeated Path Planning



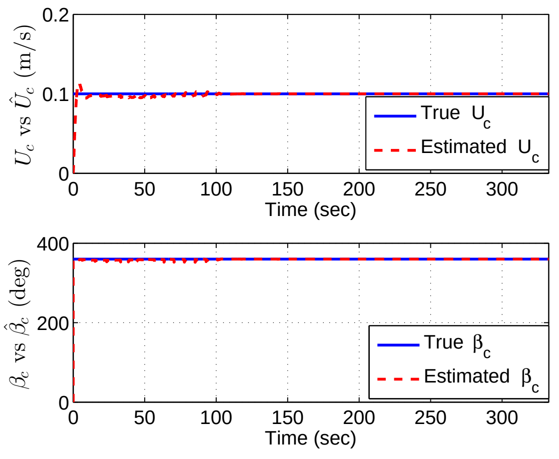
(a) y_e vs. $\hat{y}_e$ and θ_y vs. $\hat{\theta}_y$



(b) x_e vs. $\hat{x}_e$ and θ_x vs. $\hat{\theta}_x$

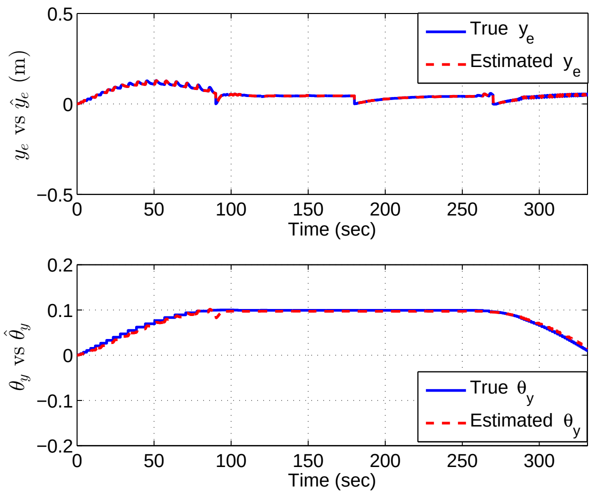


(c) u_r vs. u_{r_d} and ψ vs. ψ_d

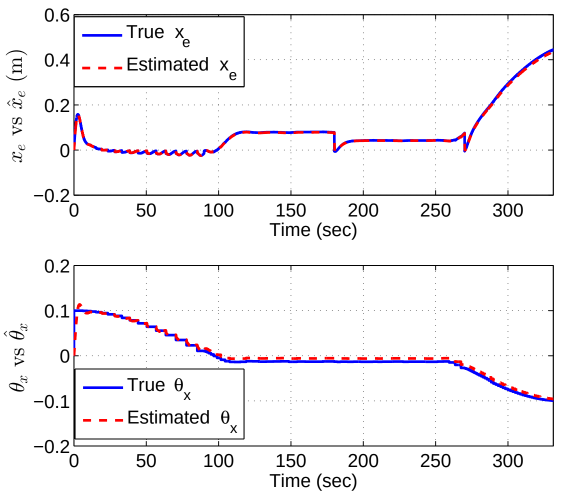


(d) U_c vs. $\hat{U}_c$ and β_c vs. $\hat{\beta}_c$

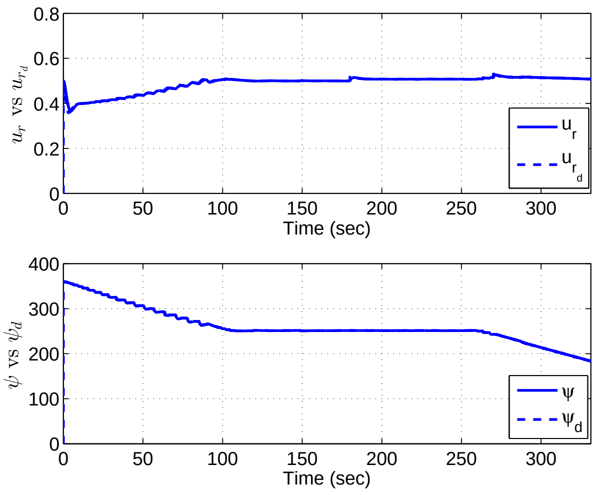
Figure 7.11: Guidance Results for Repeated Path Planning Test Case 2 with $t_{\text{new path}} = 170$



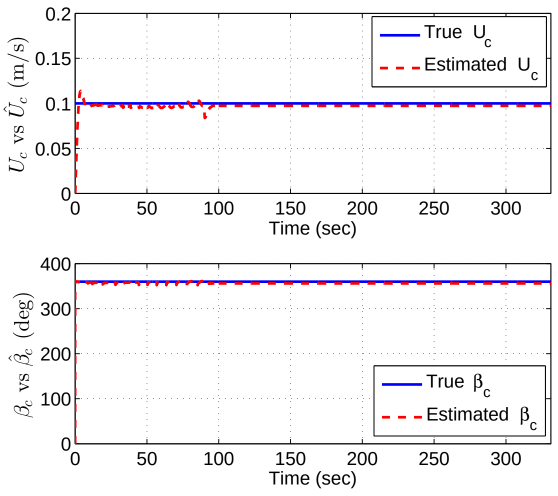
(a) y_e vs. $\hat{y}_e$ and θ_y vs. $\hat{\theta}_y$



(b) x_e vs. $\hat{x}_e$ and θ_x vs. $\hat{\theta}_x$



(c) u_r vs. u_{r_d} and ψ vs. ψ_d



(d) U_c vs. $\hat{U}_c$ and β_c vs. $\hat{\beta}_c$

Figure 7.12: Guidance Results for Repeated Path Planning Test Case 2 with $t_{\text{new path}} = 90$

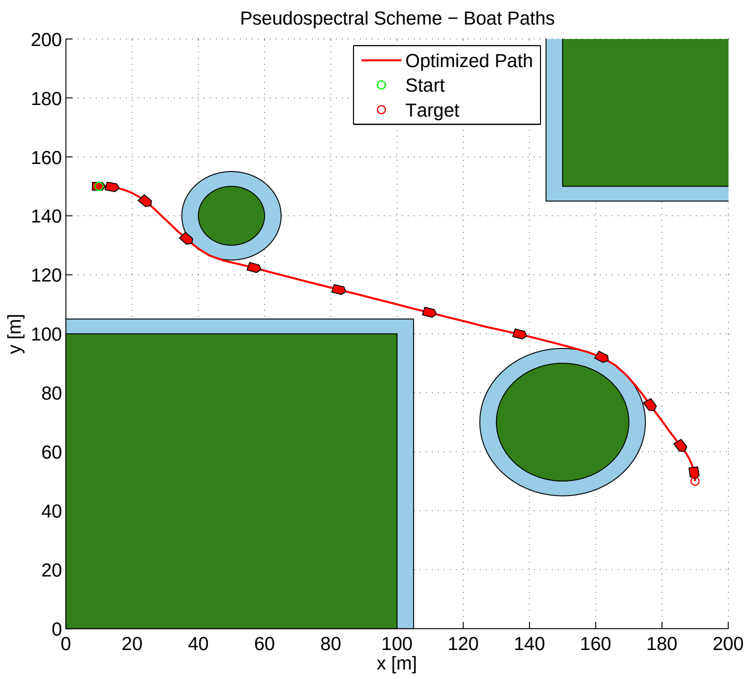
7.3 Boat Model with Ocean Current and Obstacles

In this section, the tests conducted in Section 7.2.1 and Section 7.2.2 are expanded to include an environment with ocean current and obstacles. This could, for example, represent a boat steering through a fjord where both the mainland and islands must be avoided. Another important element of this thesis, in addition to producing feasible paths, is producing safe paths. Environmental forces, such as waves, ocean currents, and gusts of wind can cause unwanted deviations from the desired path [32]. As a consequence, it is necessary to make the approach more robust by ensuring that the optimized path will always keep a minimum distance from every obstacle, thereby producing a path with a certain amount of clearance (see [20]). A security margin is therefore implemented, both to give room for the boat to pass and to have a safety margin in case the guidance system is unable to completely counteract the unknown disturbances. Table 7.2 shows the run conditions for this test.

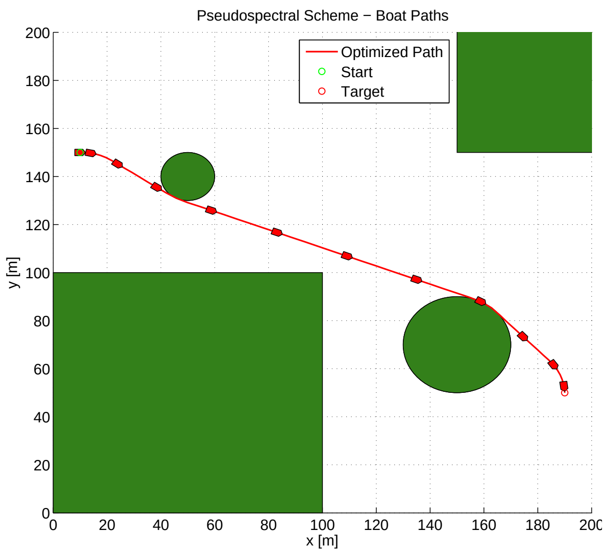
Initial pose	Final pose	ρ_{min}	Max speed	Obstacle Clearance
(10, 150, 2π)	(190, 50, $\frac{3\pi}{2}$)	20 m	0.5 m/s	5 m

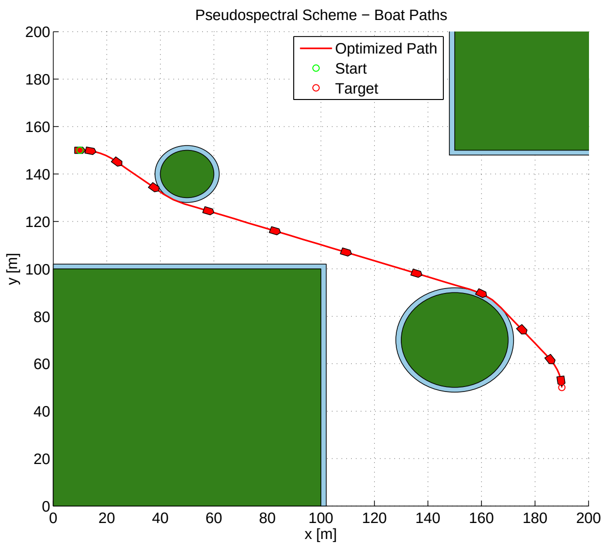
Table 7.2: Test Values for Optimized Paths for a Boat with Ocean Current and Obstacles

First, the case study is run using initial path planning and continuous current estimation, as described and tested in Section 7.2.1. In Figure 7.13, the optimized paths from this test are illustrated with different safety margins. The resulting boat path for this case study is shown in Figure 7.14, where the test is run with a safety margin of 5 meters and current $V_x = 0.0707$ m/s, $V_y = -0.0707$ m/s.



(a) Clearance = 5m

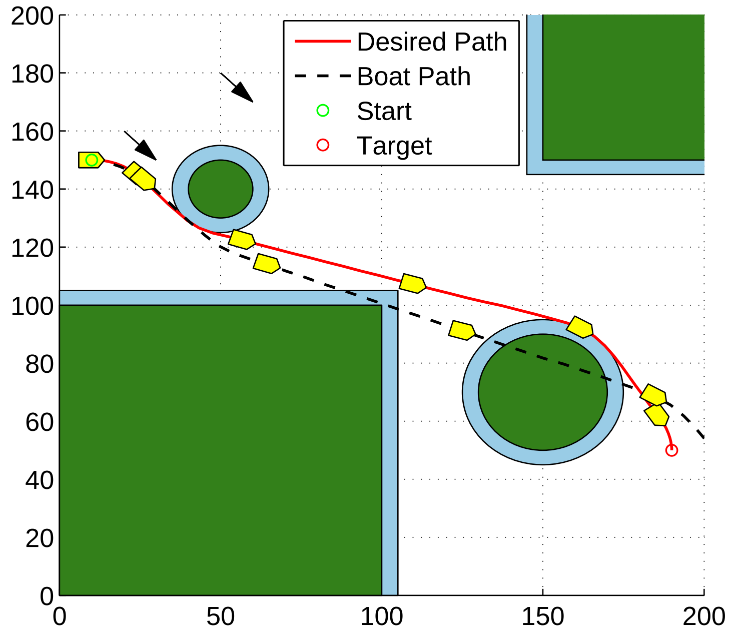




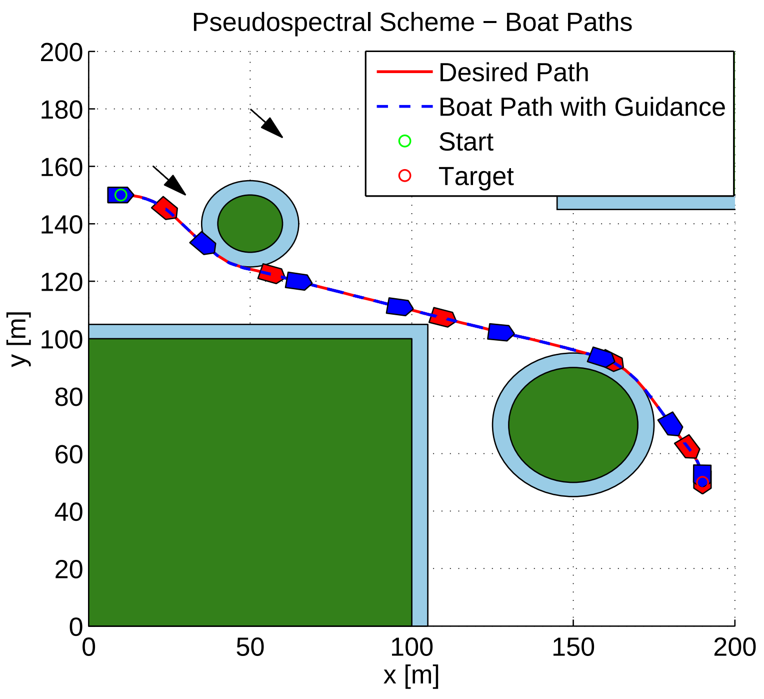
(b) No Clearance to Obstacle

(c) Clearance = 2m

Figure 7.13: Optimized Path from PS with Different Obstacle Clearance



(a) Boat Path without Adaptive Guidance



(b) Boat Path with Adaptive Guidance

Figure 7.14: Path with Obstacles Using Initial Path Planning and Continuous Current Estimation. Optimized path from $(10, 150, 2\pi)$ to $(190, 50, \frac{3\pi}{2})$

The same case study is then run with repeated path planning as described and tested in Section 7.2.2. The only change made to the environment is that the current comes from a different direction and is equal to $V_x = 0$ m/s, $V_y = -0.1$ m/s. The only reason for this change is to make it easier to illustrate the changes between the initial optimized path and the path calculated at t = 240 seconds. The optimized paths are illustrated together in Figure 7.15, while the resulting boat path for this case study is shown in Figure 7.16.

In Figure 7.15, it can be seen that both calculated paths are very similar. The only major difference is that the second estimated path (blue) has a different orientation than the first estimated path (red). This is because the second estimated path is calculated based on updated information on the ocean current. As a result, the boat changes orientation to counteract the current and to avoid crashing with the obstacles, while still taking the shortest path around the islands. In Figure 7.16, it can be seen that the boat is able to follow the desired path. After the optimized path is updated using the current estimation from the guidance system, the real boat and the optimized boat paths have the same heading.

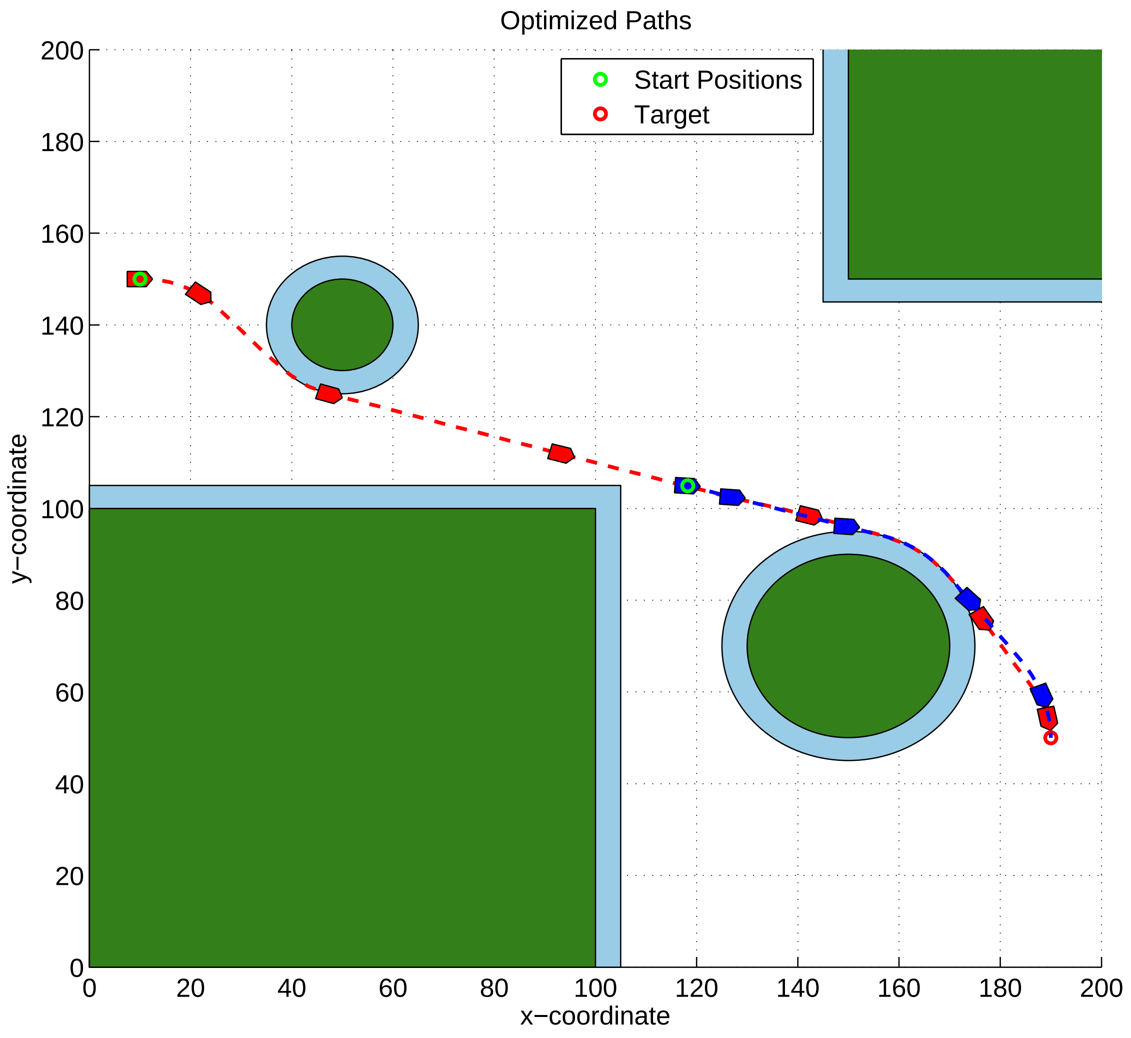


Figure 7.15: Optimized Paths with Obstacles Updated After 240 Seconds

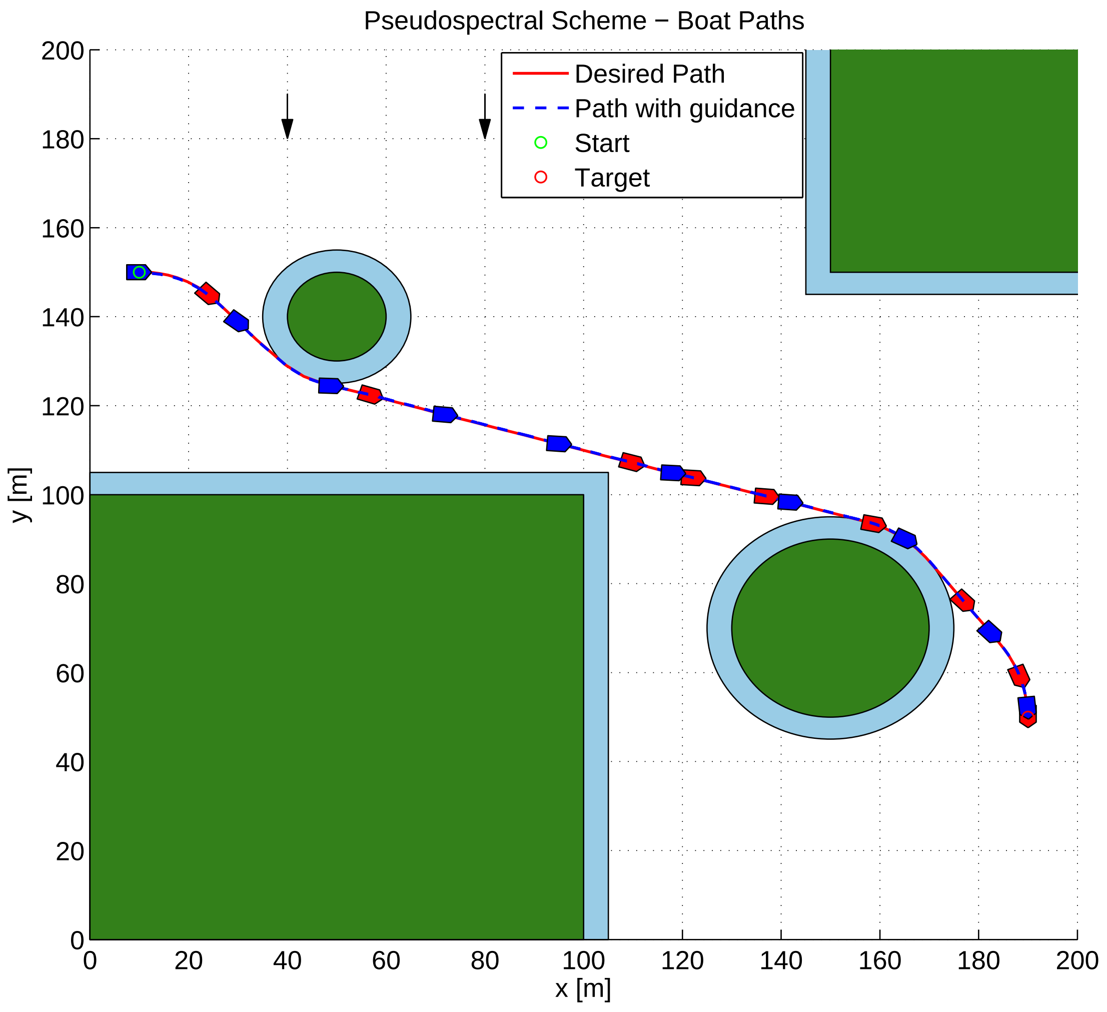


Figure 7.16: Path with Obstacles Using Repeated Path Planning. Optimized path from $(10, 150, 2\pi)$ to $(190, 50, \frac{3\pi}{2})$

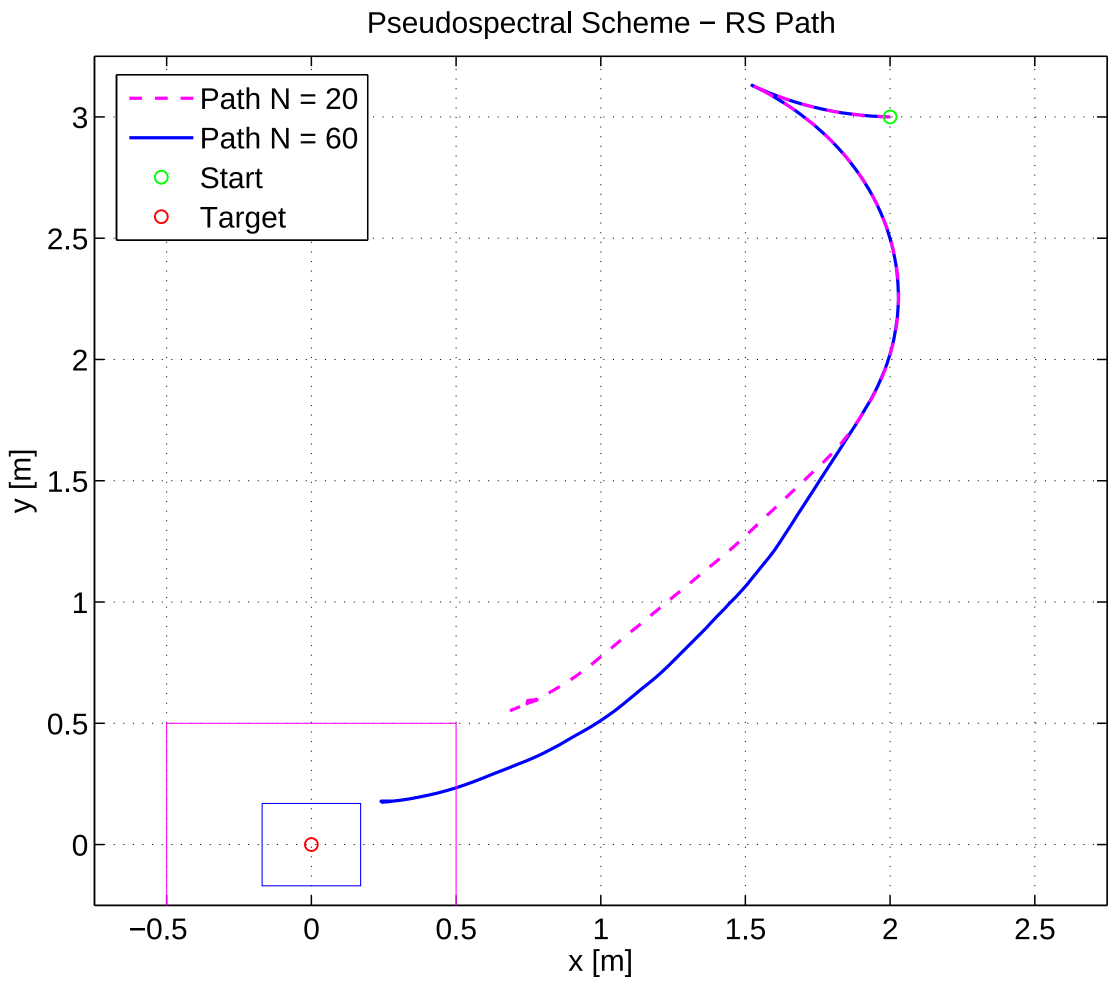
Chapter 8

Analysis and Discussion

The goal was to solve the optimal control problem and find the optimal path from one pose to another. In some cases these equations can be solved analytically, but for the models in this thesis this was not possible. As a result, numerical optimization had to be used. When using numerical optimization methods the problem is reformulated and discretised, and it therefore no longer gives the optimal solution but rather an optimized solution. In this chapter, the results from Chapter 5 and Chapter 6 are discussed and compared. In Section 8.1 and Section 8.2, each method and the results they gave for each for the various models are analysed. In Section 8.3, the two methods are compared to each other, as well as to the finite difference method studied in the pre-project for this master's thesis and to the exact solution given by Dubins paths (described in sections 4.1.1 and 4.1.2). Accuracy, advantages and disadvantages for all methods are also discussed. In Section 8.4, the results from Chapter 7 are discussed with regards to the optimized paths calculated and to the boat's ability to follow these optimized paths. Safety issues are also presented.

8.1 Semi-Lagrangian Scheme

When using numerical optimization methods the problem is reformulated and discretised, and it therefore no longer gives the optimal solution but rather an optimized solution. The optimized path is an approximate optimal path, or a path that is close to the ideal optimal path [29]. How closely related the optimal and optimized solution given by SL is, depends on the grid size, N, and refinement, h, used in the approximation. When $h \to 0$, the solution approaches the optimal path [29]. With a small N there can be large deviations between the calculated and the optimal paths. A large N gives better accuracy, but the calculation in return becomes very slow and takes a long time to converge. Figure 8.1 shows a comparison of the optimized paths for an RS car when using N = 20 and N = 60. It is seen that the larger N is for the same space, the closer it is to the optimal path given by Dubins paths. This can be seen in Table 8.1. The path lengths are about the same length, but this is because the path with N = 20 stops further away from the target. If both paths had stopped at the same distance from the target, the N = 60 path would be shorter than the N = 20 path. The run time for both tests is also shown in Table 8.1 and illustrates the major downside of this slight accuracy improvement, where the N = 60 calculations use 53 times more time to converge than the N = 20 calculations.



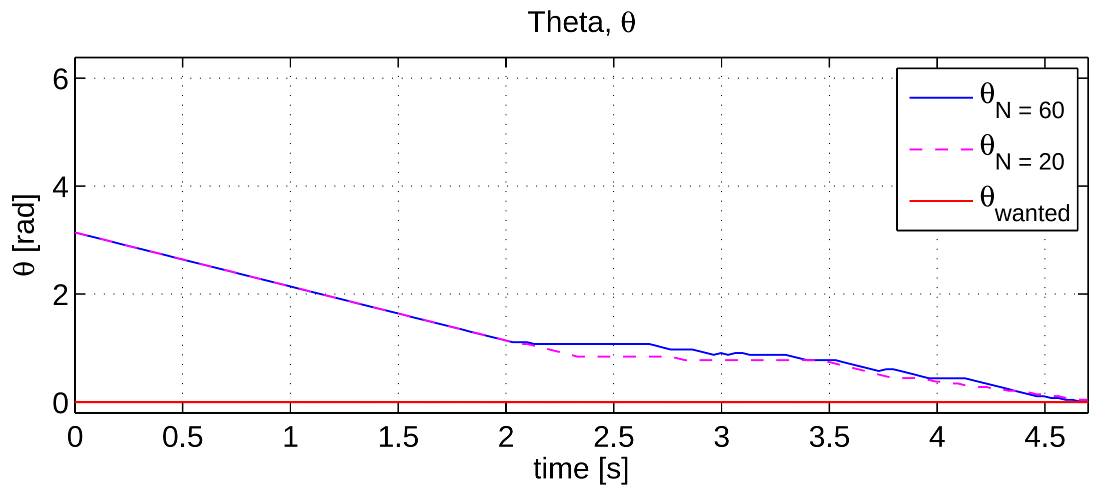


Figure 8.1: Optimized Paths Using SL with Different N Values

N	Run time	Path length	Distance from Target
20	54.7 sec	4.690 m	0.882 m
60	2904.5 sec	4.700 m	0.300 m

Table 8.1: Comparing SL Scheme with Different N Values

During the implementation of this method, some problems occurred. The vehicle does not totally reach the target, and the smaller N value used, the farther it gets from the target point. Because of the discretization and use of numerical optimization, the car finds its pose by rounding off to the closest grid node and because of low resolution it thinks it is already at the target point before it is actually there. In addition, an extended target (the target area is expanded to more than one point) has been used. This is done so that the algorithm converges more easily. This is also illustrated in Figure 8.1 where colored squares mark the expanded target area.

When examining the control inputs it can also be seen that the paths switch between steering directions more than they should when compared to the optimal Dubins paths. The small steering mistakes that appear can be explained by the discretization. In addition, the optimized control input sometimes switches back and forwards between left(u_1 = 1) and right turn(u_1 = -1), instead of going straight forward (u_1 = 0). This occurs in some degree because of the discretization, but it is also due to rounding errors in the implementation.

Another similar issue is that when the vehicle switches between driving forward and backward, it sometimes chooses (in accordance with the calculations used in this method) to alternate several times between forward and backward while taking many small turns, instead of taking one big turn and only having to switch between driving backwards and forward once. This is clearly seen in Figure 5.13. This choice would seem unnatural in the real world, but taking into consideration that these models do not account for the time needed to switch gears or stop when changing direction, it is not an unnatural choice. This seems to be an even bigger problem when at a cusp (the point were the vehicle changes directional speed between u_1 = 1 and u_1 = -1) close to the target or target orientation.

8.1.1 Car Models with Obstacles

In addition to the issues above, it has been proven that the number of nodes N and the shape of the obstacles effect the optimized path given by SL. This is seen in figures 5.6 and 5.7. Because of the discretization, a higher N and obstacles with rounded corners will in general cause SL to produce paths that go closer to the obstacles, therefore also resulting in shorter paths.

There were also additional difficulties when adding obstacles to the environment. One of the main issues was that the information did not, for all cases, spread properly through the grid. As a result, some nodes were not updated sufficiently. In the value functions illustrated in figures 5.5, 5.9 and 5.14, it is seen that the information spreads less when obstacles are present. The reason for this problem is that the value of the value function at each node is updated based on its neighbouring nodes. If the nearest node is within an obstacle (which keeps its initially high value), the information flow stops. This is why the areas in "the shadow" of an obstacle are affected most by this issue.

This problem appears more often when the obstacles are between the target and start position. As a worst case, depending on obstacle size, discretization, start and end poses, this can cause the algorithm to work improperly, causing the resulting path to deviate a great deal from the shortest path or not reach the target at all. This problem can be improved, or in some cases removed entirely, by adding more nodes and/or lowering the convergence criteria. By adding nodes the information can more easily "move around" the obstacles. Both of these fixes will, however, cause the calculation to be even slower.

8.1.2 Boat Model

Initially, the SL scheme was tested on a boat model including both surge and sway speed, thus having a model with 5 dimensions as opposed to the three in the previously tested models. The SL calculations became so slow that a decision was made to change the boat model for this test. This issue is called the "curse of dimensionality" [45] and is one of the major issues the SL scheme has. The "curse of dimensionality" is caused by the computational effort within the scheme, which grows exponentially with the dimension of the state space [46].

The second boat model did not include sway and surge speed, but included an unknown ocean current. The value function was calculated to produce optimal paths where there were no disturbances present. Consequently, the paths produced were not optimal. In addition, the unknown current prevents the boat from reaching the exact target position. Since the control input is continuously calculated with regards to current position and the value function (which always tries to pull the vehicle towards the target), and not a precalculated path, the boat manages to move towards the target without any major detours. See Figure 5.15. This is a much better result than the results achieved when using PS and an unknown ocean current, seen in Figure 7.3.

8.2 Pseudospectral Method

The Legendre pseudospectral method is another technique used to solve the optimal control problem when it is to difficult to solve analytically. The idea is to discretize the optimal control problem to cast a smooth nonlinear optimization problem, which can later be solved with, for example, an NLP solver. The pseudospectral method is also a grid-based method. The grid points are chosen at certain locations called Chebyshev nodes. When using this technique, a higher accuracy can be achieved with less grid points than with other grid-based numerical schemes. In this thesis the MATLAB application package DIDO was used in these implementations.

The PS method gives an optimized path with very high accuracy using less computational time than competing methods. In addition, the number of nodes used effects the accuracy less than when using other schemes. This can be seen in Figure 8.2. The calculations with N = 20 produce a path very similar to the calculations with N = 60, where both paths are very close to the optimal path. The only difference between the two paths in Figure 8.2 is that the path with N = 60 is slightly smoother. The reason for this is that the PS method gives the states at each node in the optimized path and then connects these nodes with straight lines. Compared to other methods, the PS scheme using DIDO still gives very accurate paths with very low computational time for both a high and low number of nodes. The path lengths and run time for both paths are shown in Table8.2.

N	Run time	Path length
20	1.58 sec	4.7742 m
60	8.33 sec	4.7497 m

Table 8.2: Comparing PS Scheme with Different N Values

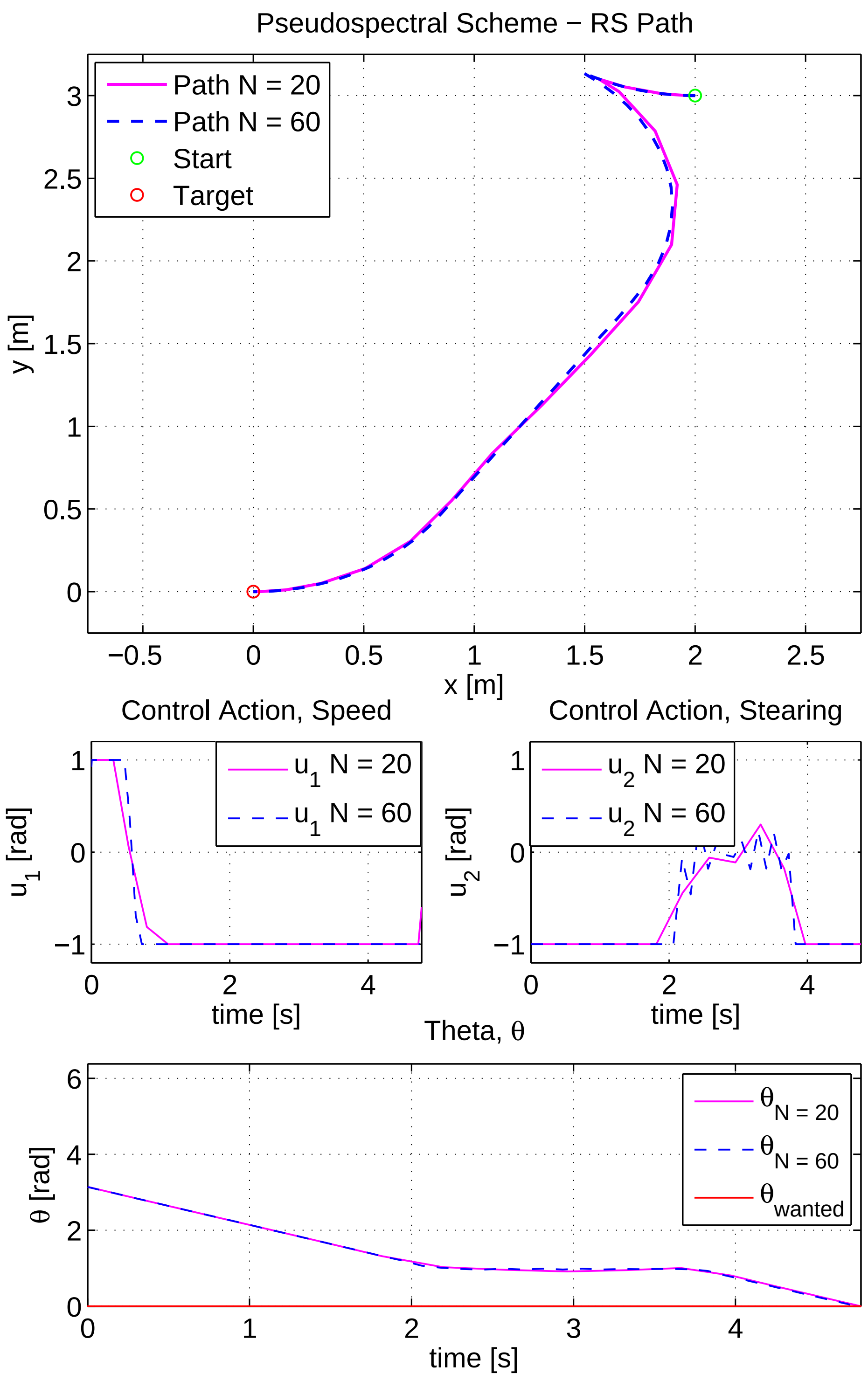


Figure 8.2: Optimized Paths Using PS with Different N Values

8.2.1 Car Models with Obstacles

When testing PS for the car models in an environment with obstacles, some problems occurred. The PS methods would sometimes produce paths that did not totally avoid the obstacles, and therefore, the paths were not feasible. When using PS and DIDO the constraints and optimality are enforced at the node points and at the specific event times. Consequently, the constraints are not always satisfied between these node points [26]. This causes the discretized solution to always be feasible, but not necessarily the continuous solution. This is one of the few drawbacks with this method, and may cause problems if there are too few nodes. One example of this is that the calculated path may cut the corner of an obstacle because the nodes are on either side of it, thereby resulting in an infeasible path. This problem is illustrated in Figure 8.3. This can be solved by introducing more nodes. The path produced is then less likely to cut the corner, but this also increases the computational complexity [36]. Another solution is to increase the obstacle size, giving it an extra "safety margin". This way the nodes must keep a greater distance to the real obstacle making it less likely for the path between the nodes to touch an obstacle. On the negative side, this forces the optimized path to be less optimal with respect to the real obstacle size. In real life situations, keeping a safe distance to obstacles in the environment is a safety issue and should always be considered and studied closely to ensure a safe path. The two fixes for corner cutting are illustrated in Figure 8.4. This type of problem will also be less likely to occur in an environment where the shape of the obstacles does not have such sharp corners [36].

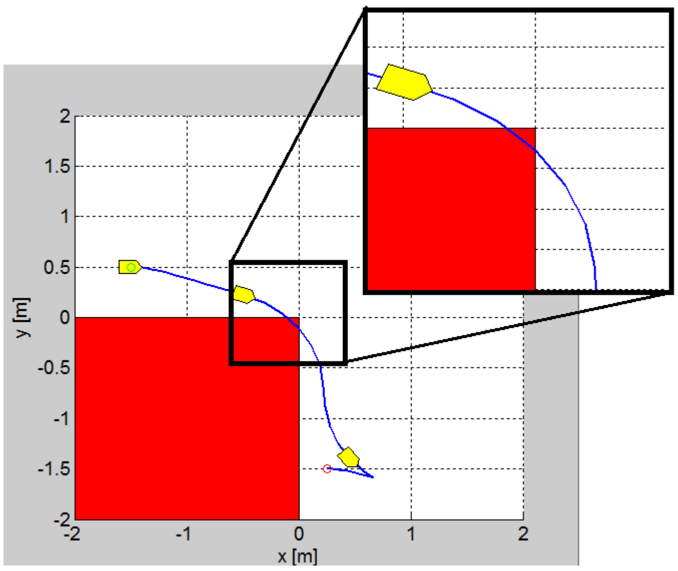
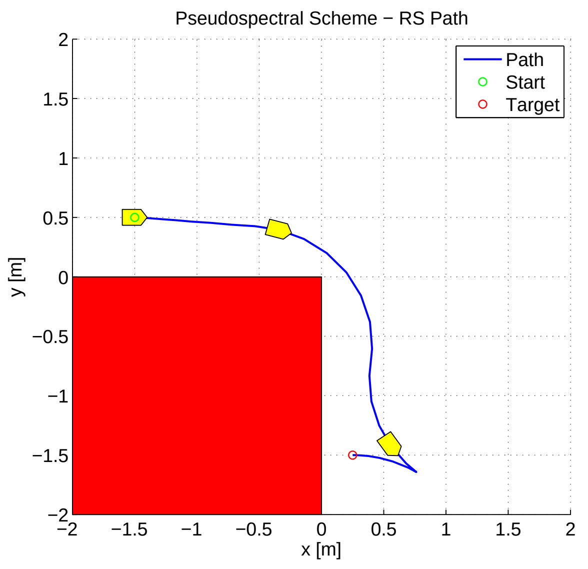
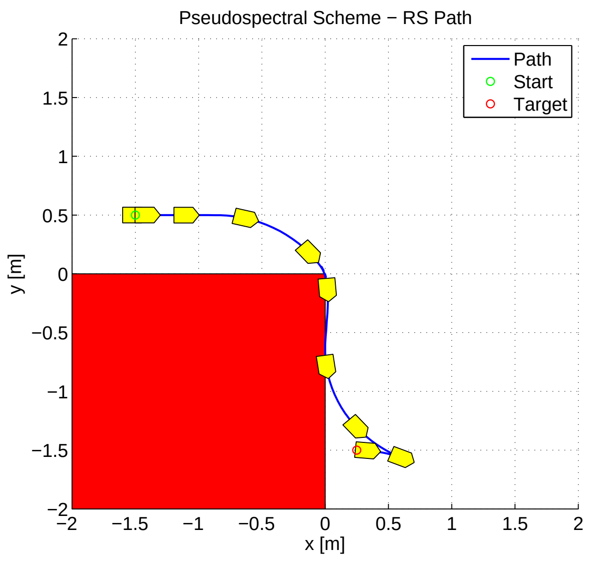


Figure 8.3: Infeasible PS Path in Obstacle Environment with N = 30



(a) Expanded Target Area with 0.2 m, N = 30



(b) Regular Target Area, N = 100

Figure 8.4: Fixing Infeasible Paths - PS with Obstacles

For some environments, especially cluttered environments including long, large and many obstacles, the PS scheme with DIDO struggles to find a feasible solution even though several feasible solutions exist. This is because DIDO generates a low-node precomputation step first, and then a more accurate high-node solution. The cause of this is that DIDO improves accuracy and computation time since the precomputation step, which is calculated based only on the initial and final position, gives a good path suggestion before the final solution is calculated [36]. However, a cluttered environment with long and large obstacles is not captured well by this low-node precomputation. This will in turn give the final computation step a very poor path estimate, thus causing the final path to be very bad or infeasible. DIDO does, however, support giving additional initial position guesses. In other words, before the first computation DIDO can be given more path positions than the initial and final position [36]. This may solve the problem in many cases. Nevertheless, given our goal of finding an optimized path from one pose to another, such a workaround contradicts the purpose since knowledge of the environment or other path positions are needed.

8.2.2 Boat Model

Unlike the SL scheme, the PS methods using DIDO easily handled the five-dimensional boat model, including both surge and sway speed. Although the PS method has a slightly longer computational time for this boat model than for the DU and RS car, it does not struggle with the "curse-of dimensionality" like most of the grid-based numerical methods do. It is easily seen in figures 6.12 to 6.14 that the optimized path takes advantage of the different directional speeds to produce the shortest path possible. In Figure 8.5, the paths for a Dubins car and the simplified boat models, given the same minimum turning radius, start and target pose, are compared. The only difference in the models is that the boat model has speed in both surge and sway, while the car only has speed in the surge direction.

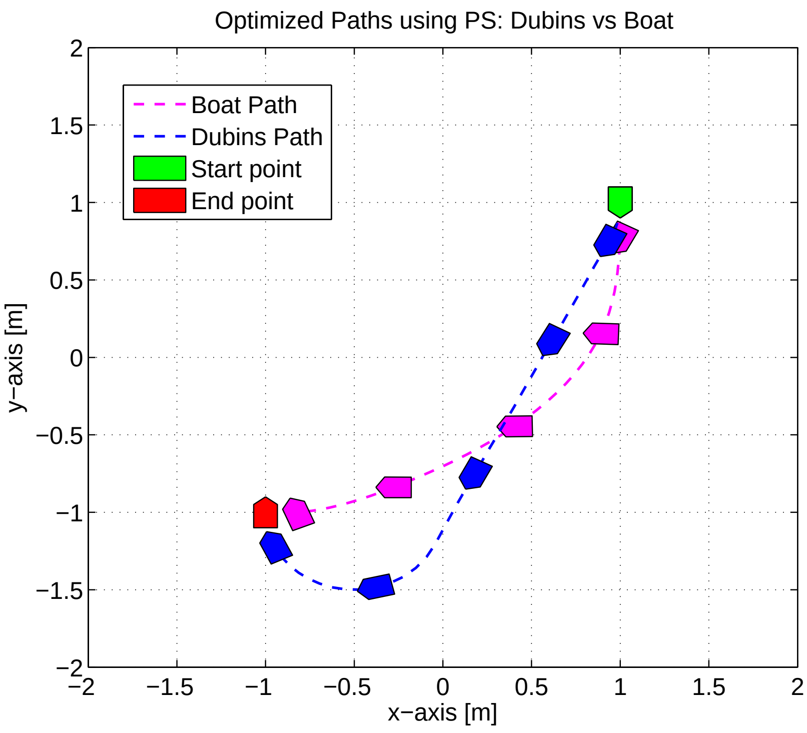


Figure 8.5: PS: Dubins vs Boat Path

It was, however, detected that the PS method using DIDO did not work well with unknown ocean currents. Since the optimized control is precalculated by DIDO, it can not react or rectify any of the position errors caused by the unknown current. It uses the calculated input controls directly. As a result, the boat drifts away from the path and is not able to reach its target. This is seen in Figure 7.3. In comparison, the SL scheme had much better results when tested with an unknown current.

8.3 Comparison of Numerical Optimization Techniques

The numerical optimization methods tested in this master's thesis are the semi-Lagrangian approximation scheme and the pseudospectral method. In addition, they were compared to the finite difference method (FD) and the optimal solution given by Pontryagins Minimum Principle (PMP), both tested in the pre-project to this master's thesis. Both SL, PS and FD are numerical optimization techniques based on dicretizing the optimal control problem before solving it. Since the problems are discretizinced, the solutions are optimized, not optimal.

Both the finite difference method and the semi-Lagrangian scheme have been used to solve the HJB equation. In more recent years, the pseudospectral (PS) method has become more popular in control theory and it has successfully been used to solve optimal control problems and nonlinear partial differential equations. Like most grid-based numerical methods, both the SL and FD method suffer from the so-called "curse-of-dimensionality", which causes the computational effort to grow exponentially with the dimension of the state space [46]. The PS method is also a grid-based method, but since the grid points are chosen at certain locations instead of the entire domain, a much higher accuracy with a smaller number of grid points can be achieved. Therefore, PS is, in general, more accurate and faster. Consequentially, it also avoids the "curse-of-dimensionality", thus allowing it to solve more complex problems. This is one major argument for choosing the PS method to solve the optimal control problem [46].

Figure 8.6 shows a comparison of the solutions given by SL, PS, FD and PMP (the optimal path) for a specific Reeds-Shepp car path from $(2, 3, \pi)$ to $(0, 0, 0)$. Table 8.3 gives the corresponding node number N, run times and path lengths. It can be seen that the path given by PS is very similar to the optimal path. Both the SL path and the FD path differ somewhat from the optimal path, though they are very similar to each other. They both have a slightly longer path. In addition, they both stop at a distance from the target. The reason for this is the discretization, rounding and enlarged targets used in these methods, making the car think it is at the target before it actually is. This is also the reason, as shown in Table 8.3, that they have shorter path lengths than the PS method, even though they do not. If all of the paths had reached the exact target point, the PS path would be the shortest path, with a path only 0.03 m (3 cm) longer than the optimal path.

Method	N	Run time	Path length
Semi-Lagrangian	40	738.03 sec	4.720 m
Pseudospectral	40	3.61 sec	4.751 m
Finite Difference	40	158.00 sec	4.500 m
Exact Solution			4.690 m

Table 8.3: Comparing Numerical Optimization Techniques

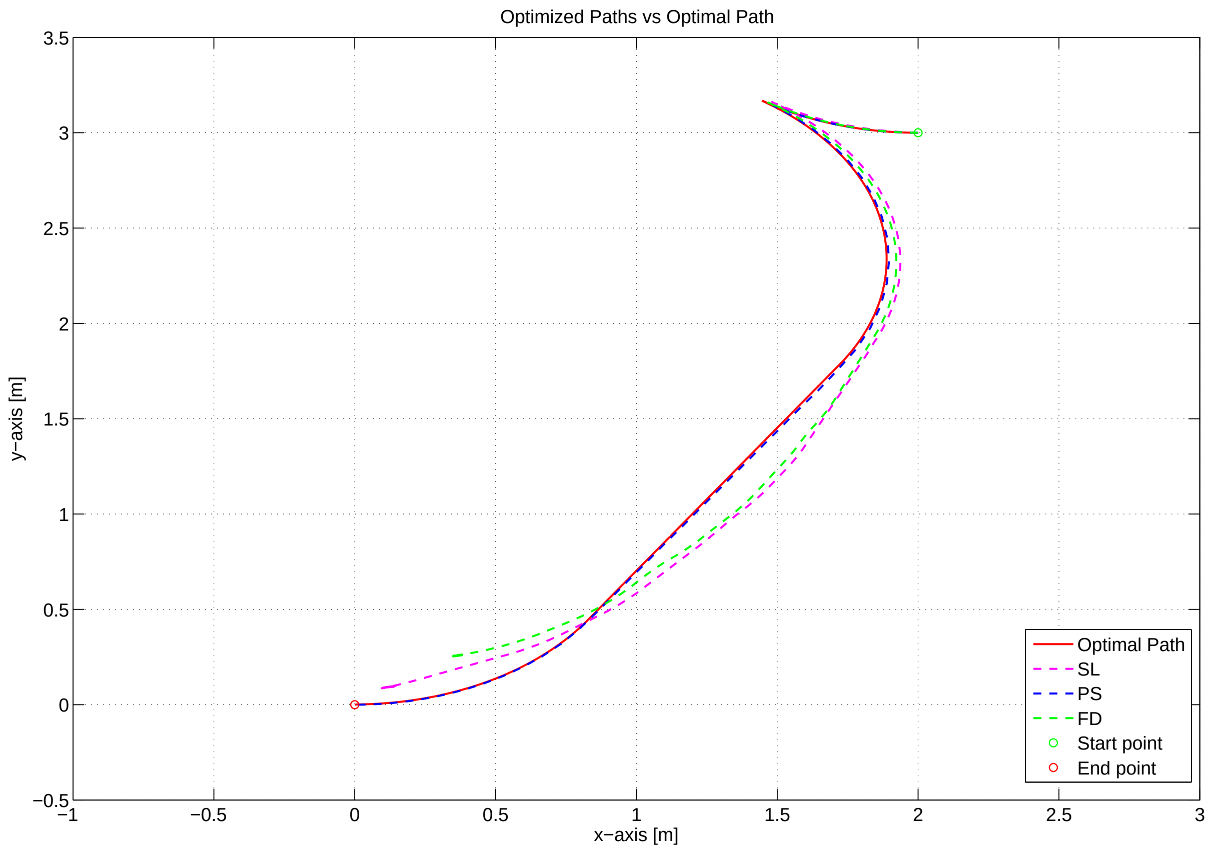
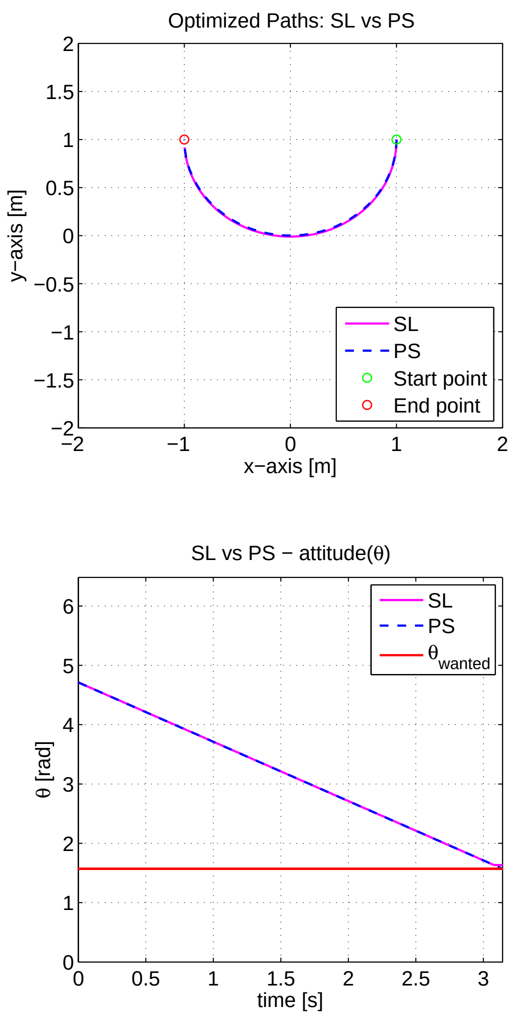


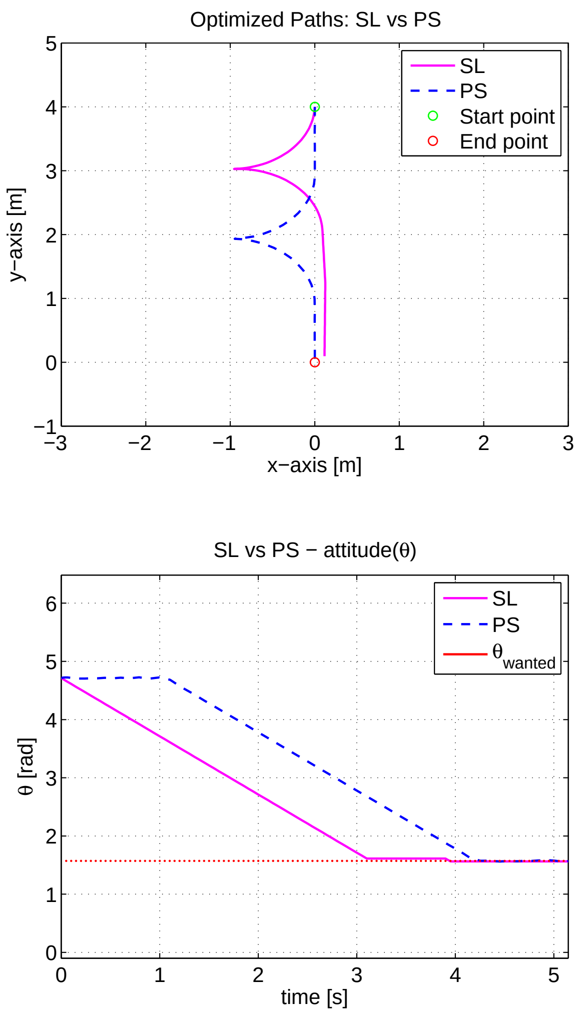
Figure 8.6: Optimized Paths vs the Optimal Path

All paths are run with N = 40. For the SL and FD scheme this results in 40 x 40 x 40 nodes spread throughout the entire domain, while the PS method only has the 40 nodes along the optimized path. Therefore, the PS method generates a much smaller-scale optimization problem when compared to the other methods. This causes the PS method to converge much faster and it avoids many of the problems and difficulties experienced when using other schemes, for example, the SL scheme [37]. The run times corresponding to the optimized path in Figure 8.6 are also presented in Table 8.3, where it is shown that PS is considerably faster than the other two methods. For this example, the FD method also converges much faster than the SL scheme. One of the main reasons for still choosing to use the semi-Lagrangian scheme over the finite difference methods, is that the SL scheme's convergence properties for discontinuous solutions are better [47]. Since some value functions are continuous and some are not, the SL scheme is capable of solving a wider range of optimal control problems.

In figures 8.7 and 8.8, several comparisons between paths given by the SL scheme and PS scheme are plotted together. For some cases, the paths are very similar, like in Figure 8.7a. In others, the paths are approximately the same length but they choose to make turns and switches at different moments along the path. This is seen in Figure 8.7b. For other start and end positions, it can be easily seen that the paths given by PS are shorter and more accurate. Based on experience, this seems to be a bigger problem when the target orientation is close to 0 or 2π. This might be because these orientations are represented at nodes near the edges of the grid, causing information not to spread as well and the discretization to have a higher influence on the control input. This can be seen in Figure 8.8, which also illustrates that the SL path is closer to the PS path when N is raised from 40 to 60.

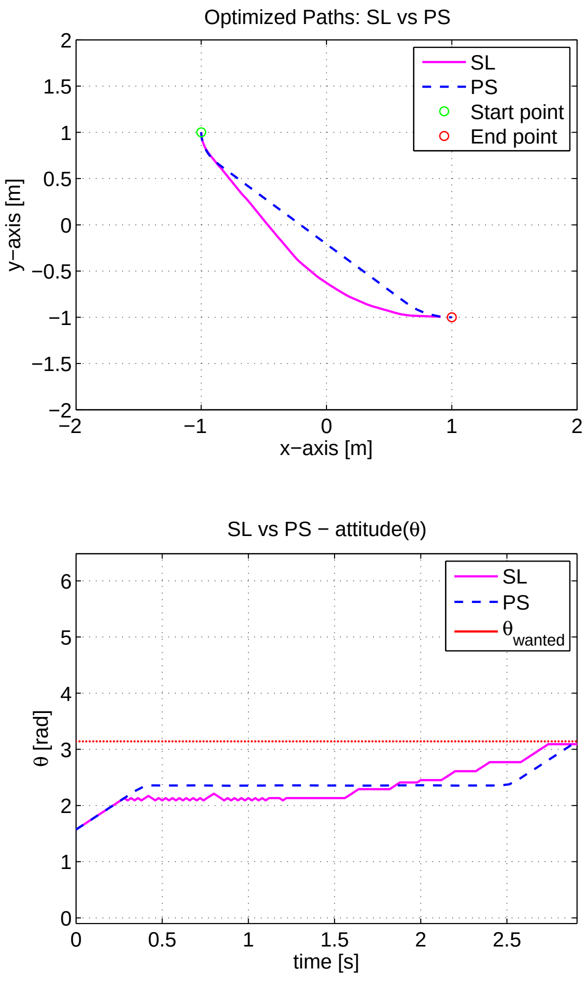


(a) Path from (1, 1, -$\frac{\pi}{2}$) to (-1, 1, $\frac{\pi}{2}$)

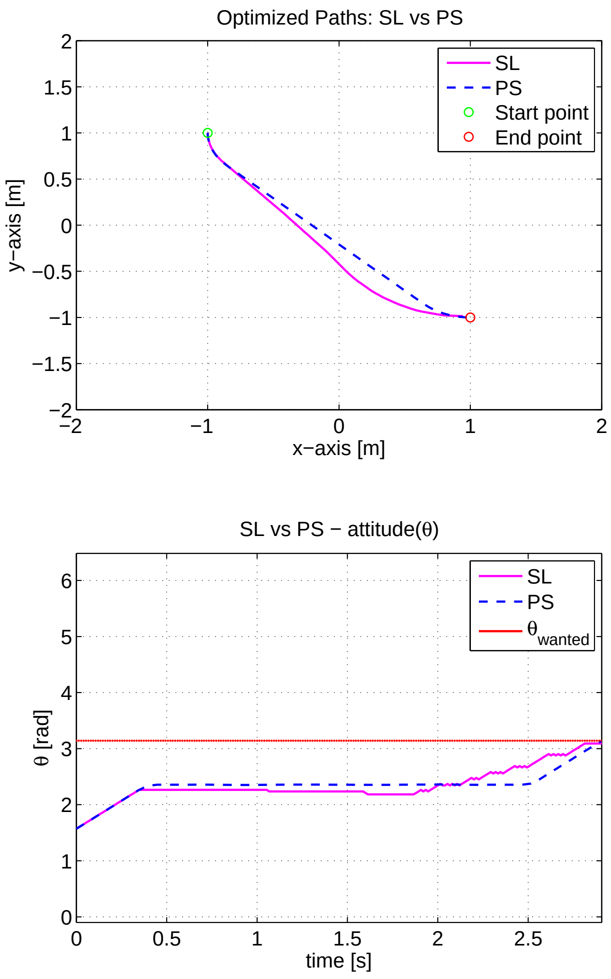


(b) Path from (0, 4, -$\frac{\pi}{2}$) to (0, 0, $\frac{\pi}{2}$)

Figure 8.7: Comparison: SL Scheme vs. PS Method



(a) SL run with N = 40



(b) SL run with N = 60

Figure 8.8: Comparison with Different N Values: SL Scheme vs. PS Method

8.4 Numerical Optimization for Path Planning and Adaptive Guidance

The considered guidance system considered uses desired positions in the x and y directions as a reference path to follow. For this thesis, these positions in time were produced using MATLAB and the MATLAB application package DIDO, which calculates an optimized path using the Legrende pseudospectral method. The guidance technique uses a combination of guidance algorithms to generate relative surge speed reference trajectories for minimizing the along-track error, and a version of the Line-Of-Sight (LOS) guidance method to minimize the cross-track error (both described in detail in [35]). By combining these guidance techniques with adaptive observers, it is possible to estimate the current components and adjust the speed and heading of the real boat in such a way that it is able to follow the desired path. Since the guidance calculation only depends on the positions given by PS and not the control input, it does not matter that the control input from PS oscillates. As long as the path given is smooth, the control input oscillation does not have any negative effects on the guidance calculations.

The guidance system and PS method were combined in two different ways. The first method tested in Section 7.2.1 uses the initial optimized path calculated by PS as the reference path during the entire simulation. This path has no knowledge of the unknown current in the environment and therefore assumes no current. In figures 7.4 and 7.5 it is seen that the guidance system updates the heading and speed in such a way that the boat is able to follow the path, reach the target position and not get pulled away by the current. A negative aspect of this method is that the guidance system has to continuously adjust the heading and speed to be able to follow the desired path, since it is impossible to follow with the desired heading and speed. As a consequence, the boat does not always reach the target with desired target orientation as the heading differs from the optimized path.

The second approach, called repeated path planning and shown in Section 7.2.2, is initialized in the same manner as the previous approach but it takes advantage of the current estimate given by the guidance system to calculate new optimized paths at a regular time interval. The initial optimized reference path is again calculated based on the assumption of no ocean current. Therefore, the guidance system has to compensate for this current by adjusting the heading and speed of the boat. After a given time interval, a new path is calculated based on the current position of the boat and the target. This path uses the ocean current estimations given by the observers in the guidance system to produce a better and more feasible path. The idea is that for each path update the estimate will be better and therefore the corresponding optimized path will be better and more feasible. During the implementation of this approach some numerical issues arose, which must be further investigated. These issues prevented the along-track and cross-track error from converging completely to zero, which they should for a well-working guidance system.

It is seen in Figure 7.10 that this works well and that the path is updated at different time intervals, thereby resulting in a different total path. When the path is updated with more information about the current, it becomes more feasible and the guidance system does not have to do as many adjustments to be able to follow the path. While the desired and real path have different orientations in the beginning, they have approximately the same orientation towards the end of the path. As a result, the boat is not only able to counteract the current and reach the target position, but it also has the correct target orientation when it reaches the target. This is an improvement over the initial path planning approach discussed above. Because of the uncertainties around the convergence of the estimated ocean current and the small deviations in the along-track and cross-track errors, initial path planning with continuous current estimate would be, however, the safest approach to use.

For both of these approaches it is, in general, slightly easier to follow the desired path illustrated in Figure 7.4. The reason for this is that the orientation of the ocean current with respect to the boat path is always the same and it is therefore easier for the adaptation algorithm to estimate the unknowns. The guidance system will give better and more consistent results when the path is smooth. In other words, a curved path presents a greater challenge than a straight line, and a curved path with sharp turns presents a greater challenge than a path where the turns have a large turning radius.

Figure 8.9 shows a comparison of the optimized paths calculated using the two different approaches, as well as the optimized path calculated by PS where the ocean current is known from the beginning. The path calculated with repeated path planning is updated at t = 170 seconds. It can be seen that after the update point at approximately (x, y) = (35, 70), the updated path and path with known current have the same heading and approximately the same path. In addition, both converge towards a point at about (x, y) = (22, 25) where the paths start to curve.

In Section 7.3 the approaches discussed above were tested in an environment including obstacles. Again, both worked well. For the first approach, the guidance system has to once again adjust the heading and speed to counteract the unknown current during the entire run. For the second approach, the heading and speed of the boat converge toward the heading and speed of the optimized path after the update at t = 240 seconds. In situations like this, it is very important to consider the safety issues. It is important to keep a safe distance to obstacles in order to have room for the physical vehicle itself in addition to a safety margin in case of malfunctions. The guidance on board could fail, the ocean current estimations could be wrong or the ocean current might just be too large for the boat to counteract. It is therefore extremely important to have sufficient time and space to react and a safety margin such that small position errors do not cause a crash.

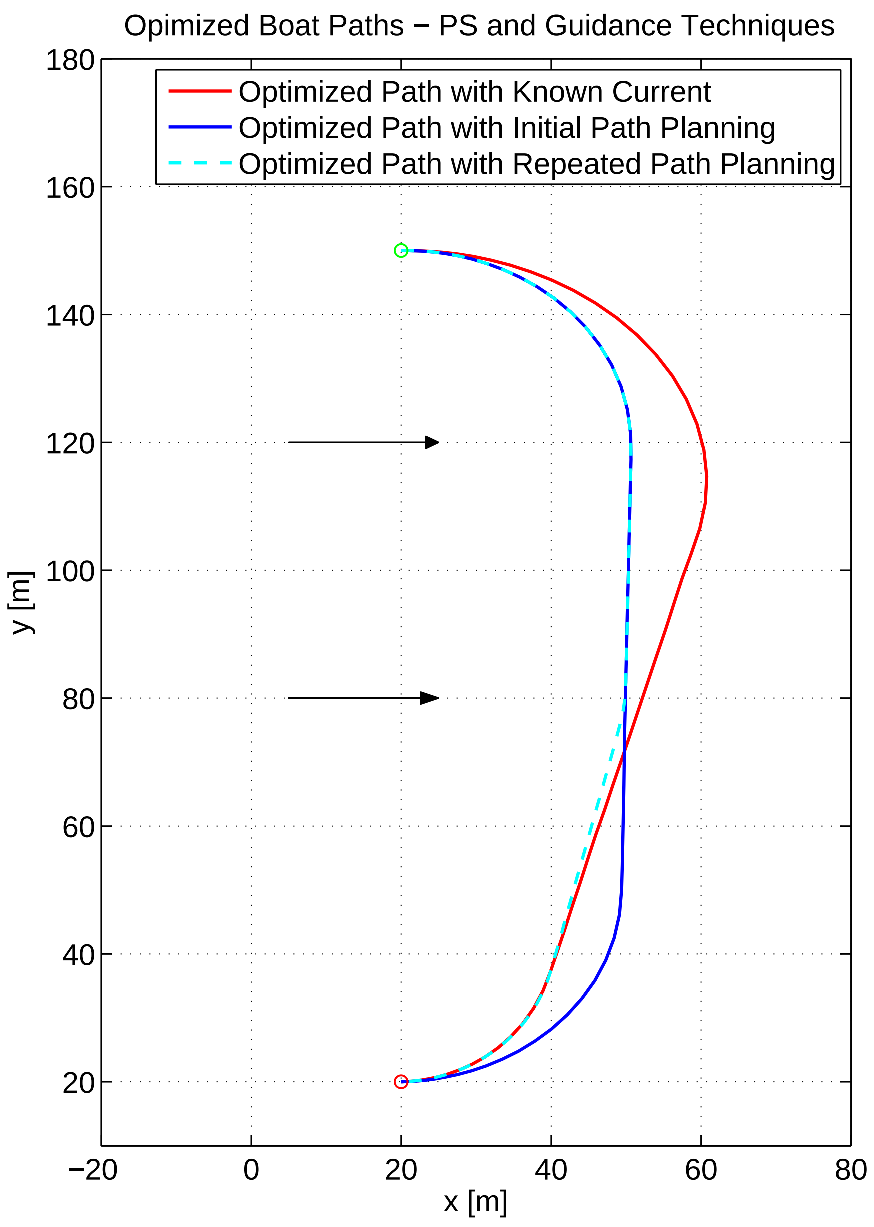


Figure 8.9: Optimized Paths Given by PS and Different Guidance Approaches

Chapter 9

Conclusion and Further Work

In this chapter the main results and conclusions from this thesis are presented and further work discussed.

9.1 Concluding Remarks

Path planning is an important part of autonomous systems and finding the optimal paths for various different vehicles is an important part of many applications. The aim of this thesis has been to find the shortest paths from one pose to another using optimal control theory and the numerical optimization methods. The numerical methods tested were the semi-Lagrangian (SL) approximation scheme and the pseudospectral (PS) method. Both methods were tested on the Dubins car, the Reeds-Shepp car and simplified boat models. Using the SL scheme to calculate the optimized path for a simplified boat model was one of the contributions in this thesis.

Since the optimization problem was solved with numerical optimization methods, the resulting paths were optimized, but not optimal. How closely related the paths from SL and PS were to the optimal paths depended on the discretization. Both schemes gave good approximations to the shortest path problem. It was proven, however, that the PS method was more computationally efficient and had greater accuracy than the paths given by SL. One of the reasons for this was the choice of grid nodes in the different methods. The SL and FD, like most grid-based numerical methods, both suffered from the so-called "curse-of-dimensionality", which causes the computational effort to grow exponentially with the dimension of the state space [46]. The PS method is also a grid-based method, but the grid points were chosen at certain locations instead of throughout the entire domain. Therefore, it had to solve a much smaller-scale optimization problem than SL and FD [37]. Consequently, it was, in general, more accurate and faster, thereby avoiding the "curse-of-dimensionality".

The PS method proved to be both very efficient and accurate. As a result, a main contribution of this thesis was to further investigate the PS method by combining it with adaptive guidance techniques to estimate and remove errors caused by unknown ocean current. It was shown that adaptive observers and a guidance system could be used to estimate unknown disturbances like ocean current, and that this estimate could be used to either minimize position error or estimate new and more feasible optimized paths that take ocean current into consideration. Although both approaches worked well, the initial path planning was preferred as a result of the uncertainties surrounding the convergence of the ocean current estimate and the numerical issues preventing the along-track error from converging to zero.

9.2 Further Work

This master's thesis was carried out over a period of 5 months in the spring of 2015. When spending this much time on a problem, several areas of possible improvements, extensions and other research ideas arise.

The tests done in this master's thesis have proved that the numerical optimization methods work and that it is possible to find the shortest paths for different models. More testing could still be done. For the SL scheme it would be interesting to test with finer refinements, to determine whether improvements can be made and to see how close these optimized paths can get to the optimal paths. The effects such improvements have on the computational time could also be tested to determine whether the improvements are worth it. The tests could be run on a more powerful computer, or multiple computers solving parts of the problem in parallel, so it would be possible to test systems with more dimensions and determine the feasibility of calculating better refinements.

In this thesis, SL was run with linear interpolation and forward Euler approximation scheme. Therefore, another subject to investigate is whether the results could be improved with cubic or quintic interpolation and/or a more complex approximation scheme like Heun (H) or Crank-Nicolson (CN). Since the PS method has proved to be both efficient and accurate, it would be interesting to test this method with models using a higher complexity, for example 3D models of UAVs and AUVs, and see how well this is handled.

There are several extensions that can be done to the models and environment tested in this thesis. One extension, which makes the problem more relevant and realistic, is to check the possibility of expanding the environment to include more realistic obstacles, both in number and different shapes and moving objects. Some extensions to the models themselves could be more variables and states, or removing restrictions and constraints, for example, constant speed. By having varying speed, the fastest way to alter the course would no longer necessarily be turning with minimum turning radius, because the smaller the turning radius, the slower the vehicle will have to move to make the turn. By introducing more complex models and more realistic environments, the situations and paths would be more comparable to real world situations.

There are many other methods besides optimization techniques for finding the shortest paths. Researching these will give better information about the differences, advantages and disadvantages for the different approaches. This information is necessary in order to be able to choose the best method for a specific situation and purpose. Optimization techniques can, as we have seen in this thesis, become very computationally expensive. Discussing their benefits and drawbacks compared to other techniques is an interesting and important subject for future work.

In this thesis, the possibility of combining numerical optimization for vehicle path planning with adaptive guidance for unknown ocean current compensation was investigated. For the repeated path planning approach, some numerical issues arose. Further work should be done to solve these issues. In addition, an attempt was made to implement repeated path planning without convergence testing of the current estimate, thereby letting the guidance system continuously estimate the current for the entire run. This implementation also had numerical issues regarding the estimate of the current component θ_x. This led to the decision to implement with the convergence test of the current estimate. Fixing these numerical issues, and thus producing an implementation with repeated path planning and continuous current estimation, is therefore another element for further work. Another extension is to implement with other numerical optimization methods or different path planning techniques.

Bibliography

[1] A. Alla, M. Falcone, and D. Kalise. An efficient policy iteration algorithm for dynamic programming equations. *SIAM Journal on Scientific Computing*, 37(1):A181–A200, 2015.

[2] D. A. Anisi. Optimal motion control of a ground vehicle. *Swedish Defense Research Agency, Tech. Rep*, 2003.

[3] J. Boissonnat, A. Cérézo, and J. Leblond. A note on shortest paths in the plane subject to a constraint on the derivative of the curvature. Technical report, INRIA, Sophia-Antipolis, Nice, France, 1994.

[4] J. C. Bowman, M. A. Yassaei, and A. Basu. A fully Lagrangian advection scheme. *Journal of Scientific Computing*, 64(1):1–27, 2014.

[5] M. Breivik. *Topics in guided motion control of marine vehicles.* PhD thesis, Norwegian University of Science and Technology, 2010.

[6] M. Breivik and T. I. Fossen. Path following for marine surface vessels. In *Proceedings of the OCEANS'04. MTTS/IEEE TECHNO-OCEAN'04*, volume 4, pages 2282–2289, Kobe, Japan, 2004.

[7] M. Breivik, V. E. Hovstein, and T. I. Fossen. Straight-line target tracking for unmanned surface vehicles. *Modeling, Identification and Control*, 29(4):131–149, 2008.

[8] H. Chitsaz and S. M. LaValle. Time-optimal paths for a Dubins airplane. In *Proceedings of the Decision and Control, 2007 46th IEEE Conference*, pages 2379–2384, 2007.

[9] H. M. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of robot motion: Theory, algorithms, and implementation.* MIT Press, 2005.

[10] R. Courant, E. Isaacson, and M. Rees. On the solution of nonlinear hyperbolic differential equations by finite differences. *Communications on Pure and Applied Mathematics*, 5(3):243–255, 1952.

[11] E. Cristiani. *Fast Marching and Semi-Lagrangian Methods for Hamilton-Jacobi Equations with Applications.* PhD thesis, Sapienza - Università di Roma, 2005-2006.

[12] N. S. Dattani. Linear multistep numerical methods for ordinary differential equations. *ArXiv preprint arXiv:0810.4965*, 2008.

[13] M. Diehl. Optimization: An overview. Lecture notes from Numerical Optimal Control Course Trondheim, Norway, available at `http://homes.esat.kuleuven.be/~mdiehl/TRONDHEIM/`, June 2010. [Online; accessed 18-August-2014].

[14] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[15] L. C. Evans. An introduction to mathematical optimal control theory version 0.2. Lecture notes from University of California Berkeley, available at `http://math.berkeley.edu/~{}evans/control.course.pdf`, 1983. [Online; accessed 10-October-2014].

[16] M. Falcone and R. Ferretti. *Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations.* SIAM, 2013.

[17] T. I. Fossen. *Handbook of marine craft hydrodynamics and motion control.* John Wiley & Sons, 2011.

[18] T. I. Fossen, M. Breivik, and R. Skjetne. Line-Of-Sight path following of underactuated marine craft. *Proceedings of the 6th IFAC MCMC, Girona, Spain*, pages 244–249, 2003.

[19] T. I. Fossen and A. M. Lekkas. Direct and indirect adaptive integral line-of-sight path-following controllers for marine craft exposed to ocean currents. *International Journal of Adaptive Control and Signal Processing*, 2015.

[20] R. Geraerts and M. H. Overmars. Clearance based path optimization for motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2386–2392. IEEE, 2004.

[21] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 2002.

[22] Q. Gong, F. Fahroo, and I. M. Ross. Spectral algorithm for pseudospectral methods in optimal control. *Journal of Guidance, Control, and Dynamics*, 31(3):460–471, 2008.

[23] P. Hamill. *A Student's guide to Lagrangians and Hamiltonians.* Cambridge University Press, 2013.

[24] R. Heinzl. *Concepts for scientific computing.* PhD thesis, Technical University of Vienna, Austria, Faculty of Electrical Engineering and Information Technology, 2007.

[25] D. Houcque. Applications of MATLAB: Ordinary differential equations (ODE). Lecture note from Robert R. McCormick School of Engineering and Applied Science, Northwestern University Evanston, available at `https://www.mccormick.northwestern.edu/docs/efirst/ode.pdf`, 2008. [Online; accessed 20-November-2014].

[26] J. P. How. DIDO - nonlinear optimization. Lecture notes from Massachusets Institute of Technology, Cambridge, Massachusetts, available at `http://dspace.mit.edu/bitstream/handle/1721.1/45583/16-323Spring-2006/OcwWeb/Aeronautics-and-Astronautics/16-323Spring-2006/LectureNotes/index.htm`, 2006. [Online; accessed 28-May-2015].

[27] H. Igel. Pseudospectral methods. Lecture notes from Ludwig-Maximilians University München, available at `http://www.geophysik.uni-muenchen.de/~igel/Lectures/NMG/04_pseudospectralmethods.ppt`. [Online; accessed 21-May-2015]].

[28] W. Jiang. Dubin vehicle path planning system. Technical report, Institute for Systems Research, University of Maryland, 2008. Available at `http://www.eng.umd.edu/~austin/ense623.d/projects08.html`, [Online; accessed 10-October-2014].

[29] J.-P. Laumond, editor. *Robot Motion Planning and Control.* Springer, 1998.

[30] J.-P. Laumond, N. Mansard, and J.-B. Lasserre. Optimality in robot motion: Optimal versus optimized motion. *Communications of the ACM*, 57(9):82–89, 2014.

[31] S. M. LaValle. *Planning algorithms.* Cambridge University Press, 2006.

[32] A. M. Lekkas. *Guidance and Path-Planning Systems for Autonomous Vehicles.* PhD thesis, Norwegian University of Science and Technology, 2014.

[33] A. M. Lekkas and T. I. Fossen. Integral LOS path following for curved paths based on a monotone cubic hermite spline parametrization. *IEEE Transactions on Control Systems Technology*, 22(6):2287–2301, 2014.

[34] A. M. Lekkas and T. I. Fossen. Minimization of cross-track and along-track errors for path tracking of marine underactuated vehicles. In *Proceedings of the IEEE Control Conference (ECC)*, pages 3004–3010, 2014.

[35] A. M. Lekkas and T. I. Fossen. Trajectory tracking and ocean current estimation for marine underactuated vehicles. In *Proceedings of the 2014 IEEE Conference on Control Applications (CCA)*, pages 905–910, 2014.

[36] L.-P. R. Lewis. *Rapid motion planning and autonomous obstacle avoidance for unmanned vehicles.* PhD thesis, Monterey, California. Naval Postgraduate School, 2006.

[37] L.-P. R. Lewis, I. M. Ross, and Q. Gong. Pseudospectral motion planning techniques for autonomous obstacle avoidance. In *Proceedings of the 2007 46th IEEE Conference on Decision and Control*, pages 5997–6002, 2007.

[38] D. Liberzon. *Calculus of variations and optimal control theory: A concise introduction.* Princeton University Press, 2011.

[39] J. Luis Sanchez-Lopez and P. Campoy. Ship deck simulation for autonomous landing of VTOL RPAS. Available at `http://138.100.76.11/visionguided2/?q=node/327`, [Online; accessed 28-February-2015].

[40] M. Owen, R. W. Beard, and T. W. McLain. Implementing Dubins airplane paths on fixed-wing UAVs*. In *Handbook of Unmanned Aerial Vehicles*, pages 1677–1701. Springer, 2014.

[41] K. Y. Pettersen and H. Nijmeijer. Underactuated ship tracking control: Theory and experiments. *International Journal of Control*, 74(14):1435–1446, 2001.

[42] A. V. Rao. Pseudospectral methods for optimal control: Theory and practice. Lecture note from Department of Mechanical and Aerospace Engineering, University of Florida, available at `http://www.eng.ox.ac.uk/control/events/pseudospectral-methods-for-optimal-control-theory-and-practice`. [Online; accessed 29-April-2015].

[43] J. Reeds and L. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[44] I. M. Ross. A beginner's guide to DIDO: A MATLAB application package for solving optimal control problems. *Ellisar, LLC, Monterey, CA*, 2007, 1998.

[45] I. M. Ross and F. Fahroo. Legendre pseudospectral approximations of optimal control problems. In *New Trends in Nonlinear Dynamics and Control and their Applications*, pages 327–342. Springer, 2003.

[46] W. Song and S. J. Dyke. Application of pseudospectral method in stochastic optimal control of nonlinear structural systems. In *Proceedings of the IEEE American Control Conference (ACC)*, pages 2504–2509, 2011.

[47] R. Takei and R. Tsai. Optimal trajectories of curvature constrained motion in the Hamilton-Jacobi formulation. *Journal of Scientific Computing*, 54(2-3):622–644, 2013.

[48] L. Techy and C. A. Woolsey. Minimum-time path planning for unmanned aerial vehicles in steady uniform winds. *Journal of Guidance, Control, and Dynamics*, 32(6):1736–1746, 2009.

[49] The Open Motion Planning Library. Path planning. Picture available at `http://ompl.kavrakilab.org/`. [Online; accessed 30-May-2015].

[50] A. Tsourdos, B. White, and M. Shanmugavel. *Cooperative path planning of unmanned aerial vehicles.* John Wiley & Sons, 2010.

[51] Wikipedia. Richard Ernest Bellman. Picture available at `http://en.wikipedia.org/wiki/Richard_E._Bellman`. [Online; accessed 28-May-2015].